VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

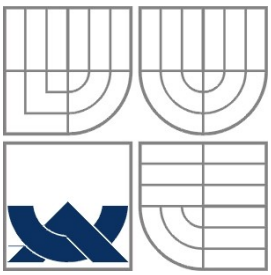
TVORBA APLIKACE PRO ZOBRAZOVÁNÍ 3D OBJEKTŮ PRO PROSTOROVÉ VIDĚNÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

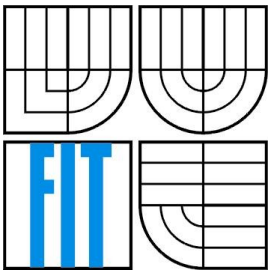
AUTOR PRÁCE
AUTHOR

FRANTIŠEK KUBIŠ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA APLIKACE PRO ZOBRAZOVÁNÍ 3D OBJEKTŮ PRO PROSTOROVÉ VIDĚNÍ

APPLICATION FOR VISUALIZATION OF 3D OBJECTS FOR STEREOSCOPIC VISION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK KUBIŠ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. TOMÁŠ MIKOLOV

BRNO 2010

Abstrakt

Tato práce se zabývá způsobem jak vytvořit stereoskopickou projekci a stereoskopický obraz na počítači, co jej ovlivňuje a jak jej dále vylepšit. Je zde popsáno několik běžně používaných zobrazovacích technologií a jsou zde teoreticky popsány knihovny využité při návrhu a implementaci práce. Poslední část se zabývá návrhem, implementací a testováním aplikace pro světla typu DigitalSpot od firmy ROBE lighting s.r.o..

Abstract

This thesis is objected to problem, how to create a stereoscopic projection and stereoscopic picture on screen, what has influence on it and how to improve it for nextime. There are described several, generally used, screening technologies and also libraries, in theoretical matter, used in designing, testing and implementation of this thesis. Last part considers about designing, iplementating and testing application of DigitalSpot light system from Robe lighting LTD.

Klíčová slova

Stereoskopická projekce, paralaxa, ghosting, off-axis, OpenGL, DigitalSpot

Keywords

Stereoscopic projection, paralax, ghosting, off-axis, OpenGL, DigitalSpot

Citace

František Kubiš: Tvorba aplikace pro zobrazování 3D objektů pro prostorové vidění, bakalářská práce, Brno, FIT VUT v Brně, 2010

Tvorba aplikace pro zobrazování 3D objektů pro prostorové vidění

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Mikolova.
Další informace mi poskytl Martin Farník.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
František Kubiš
19.5.2010

Poděkování

Na tomto místě bych rád poděkoval firmě ROBE lighting s.r.o., se kterou jsem úzce spolupracoval při tvorbě této bakalářské práce. Dále bych chtěl poděkovat Ing. Martinu Farníkovi za zastupování firmy ROBE, za rady z oblasti počítačové grafiky a stereoskopie a především za výborné rady ohledně knihovny OpenGL. A nakonec, chci poděkovat mému vedoucímu Ing. Tomáši Mikolovi za odborné vedení a rady všeho druhu.

Obsah

1 Úvod

V moderní době, kdy se počítače vyskytují téměř v každém odvětví lidské činnosti, se rozvíjí čím dál tím větší tlak na počítačovou grafiku a vzhled výsledných aplikací. Už dávno se upustilo od textového výstupu na tiskárnu, vše se nyní zobrazuje na nějakém monitoru či displeji. Tento přístup má mnoho výhod počínaje menší spotřebou – není zapotřebí žádný papír, na který se tisklo, až po větší rychlost a flexibilitu při zobrazování výsledných dat.

Dříve se veškerá data převáděla do textové podoby a následně vypisovala na displej. Tento způsob byl sice jednoduchý avšak při větším množství textu na displeji se stal nepřehledný a nepřirozený pro uživatele. Ve snaze zpříjemnit uživateli práci s počítačem se začaly hledat způsoby jak vykreslovat jednoduchou dvourozměrnou (2D) grafiku, s postupem času i trojrozměrnou (3D). V dnešní době, když se trojrozměrná grafika začíná podobat fotografiím, zjišťujeme, že to stále zdaleka není dokonalé, jelikož 3D scéna vykreslená na klasickém displeji stále vypadá ploše. Chybí zde totiž informace o hloubce, tedy o vzdálenosti objektů od pozorovatele.

Tím se dostáváme k jedné z hlavních částí této práce. Aby mohl člověk vnímat skutečnou vzdálenost objektu, musí ho vidět ze dvou různých perspektiv, jedna pro každé oko. Touto skutečností se podrobně zabývá obor zvaný stereoskopie. V další kapitole si tedy popíšeme základy stereoskopie, jak dosáhnou stereoskopického efektu a čím ho dále vylepšit. Popíšeme nejčastěji reálně používané technologie včetně anaglyfu, jejich výhody a nevýhody.

V další kapitole se pak budeme zabývat 3D grafikou, knihovnou OpenGL, SDL a lib3ds, která slouží pro načítání dat z formátu 3DS. Poté se zaměříme na vlastní řešený problém, implementaci možného řešení, experimentování a testování. V poslední kapitole si navrhneme další možná rozšíření a zhodnotíme dosažených výsledků.

2 Stereoskopie

Slovo stereoskopie je odvozeno z řeckého slova „stereo“ jenž má význam jako pevný, prostorový či tělesný. Stereoskopie pak popisuje prostorový zrakový vjem. Je důležité neplést se stereofonií, která popisuje prostorový zvukový vjem místo obrazového.

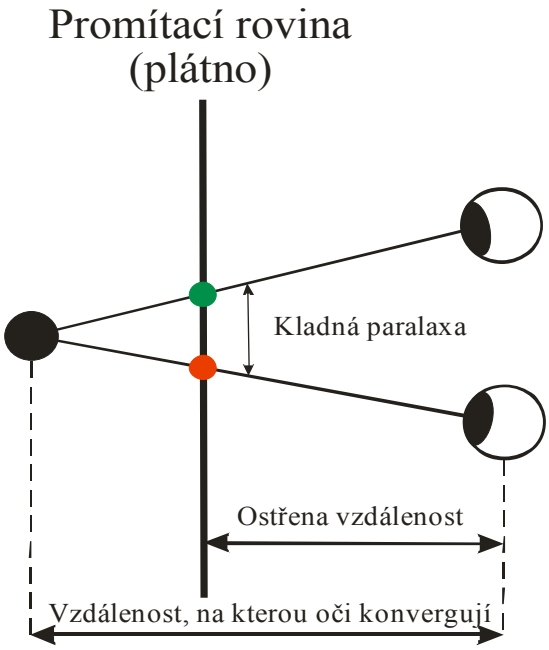
Základem stereoskopie, jak již bylo lehce zmíněno v úvodu, je fakt, že každé naše oko dostává jiné údaje – jiný obraz. Tomuto jevu říkáme binokulární disparita [1]. Dojem hloubky vzniká až v našem mozku po spojení takovýchto dvou obrazů. Tento proces spojování dvou monoskopických obrazů se někdy nazývá jako fůze nebo syntéza. Pokud by se oba obrazy od sebe vůbec nelišily (jejich binokulární disparita by byla nulová) mozek by se ani nemusel snažit je jakkoli spojit, jelikož spojením by nevzniklo nic nového. Naopak, pokud by se obrazy od sebe lišily příliš, jejich disparita by překročila dvacet úhlových minut [2], mozek je nedokáže spojit a nevznikne prostorový vjem. Musíme si ale uvědomit, že vytváříme pouze iluzi prostoru, ve skutečnosti žádný prostor ani hloubka nevzniká.

2.1 Paralaxa

V minulém odstavci bylo zmíněno, že pokud se obrazy pro levé a pravé oko neliší vůbec anebo pokud se liší příliš, nedojde k prostorovému vjemu. K tomuto tématu se neodmyslitelně váže pojem obrazová paralaxa. Máme-li nějaký objekt, u kterého chceme dosáhnout stereoskopického efektu, pak pro levé i pravé oko ho musíme zobrazit na jiném místě – z jiné perspektivy. Tato vzdálenost mezi zobrazením pro levé a pravé oko se nazývá paralaxa. Paralaxu můžeme rozdělit do několika skupin. Horizontální paralaxa udává vzdálenost na vodorovné ose x na displeji. Za normálních okolností nás zajímá pouze tato paralaxa a nebude-li řečeno jinak, budeme se dále zabývat pouze touto paralaxou. Naproti tomu stojí vertikální paralaxa, která udává vzdálenost v ose y na displeji. I malé hodnoty vertikální paralaxy mohou mít špatný vliv na výsledný stereoskopický efekt, případně efekt se může vytratit úplně. Dále můžeme paralaxu dělit podle toho, kde se bude jevit zobrazovaný předmět. Ať už se jedná o jakoukoli nenulovou paralaxu, musíme si uvědomit, že oči vždy konvergují na nějaký objekt, ale ostří na plátno. V tomto je hlavní rozdíl mezi stereoskopickým jevem a reálným světem (viz Ilustrace 1: konvergence a ostření očí). V reálném světě oči konvergují i ostří na stejnou věc.

2.1.1 Kladná paralaxa

Při využití této paralaxy se pozorovaný objekt bude jevit za promítacím plátnem. Oči tedy konvergují až za plátno. Je vhodné, aby většina celé vykreslované scény byla s touto paralaxou. Jednak oči nejsou tolik namáhány při sledování, ale hlavně nemůže dojí k rušivým vlivům jako je třeba zakrytí vystupujícího objektu okrajem promítacího plátna.



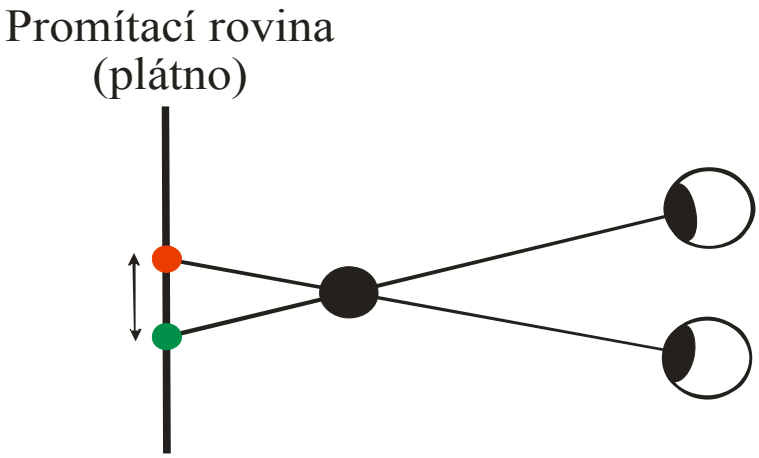
Ilustrace 1: konvergence a ostření očí

2.1.2 Nulová paralaxa

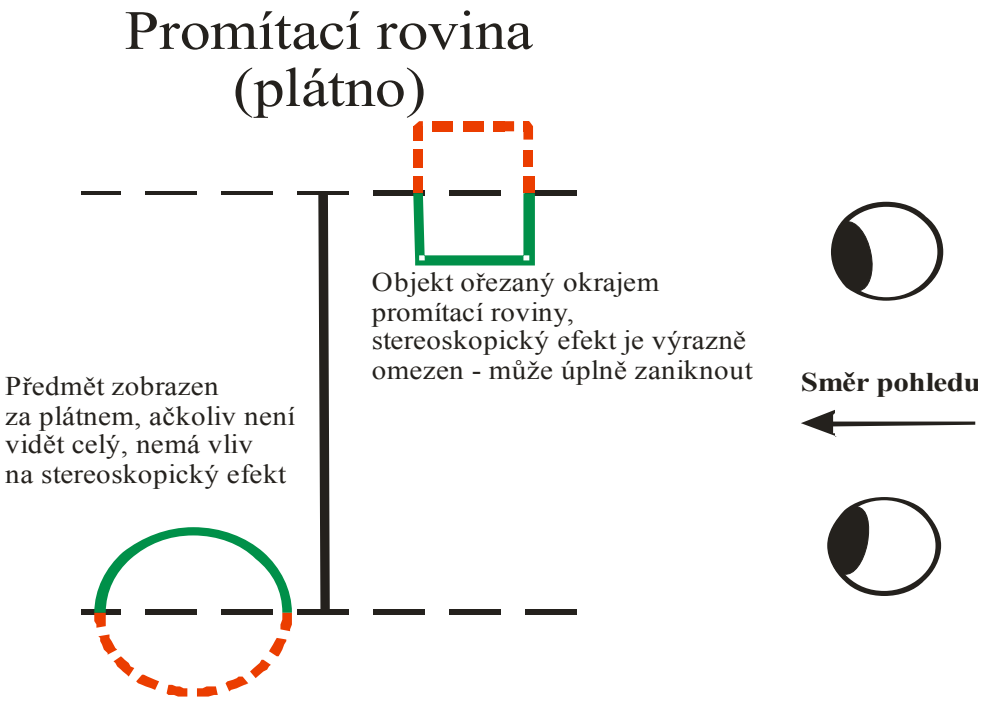
Toto je nejpřirozenější stav, pokud by všechny předměty ve scéně měly nulovou paralaxu, jednalo by se o klasický dvourozměrný obraz. Předměty s touto paralaxou se budou jevit v úrovni plátna, oči konvergují i ostří na plátno. Oči jsou nejméně namáhány.

2.1.3 Záporná paralaxa

Při využití této paralaxy (Ilustrace 2: záporná paralaxa) se zobrazovaný předmět jeví jakoby byl před plátnem. Musíme si však dávat pozor, aby byl uprostřed obrazu a nemohlo tak dojít k jeho ořezání pomocí okraje plátna. Mozek není schopen zpracovat stav, kdy objekt, který je blíž k pozorovateli, by byl zakryt objektem vzdálenějším od pozorovatele (pro názornost Ilustrace 3: ořezávání objektů). Je-li objekt zakryt horním či spodním okrajem, nepůsobí tak rušivě jako při zakrytí levým či pravým okrajem. Velké hodnoty záporné paralaxy můžeme pozorovat v různých zábavních parcích nebo kinech IMAX.



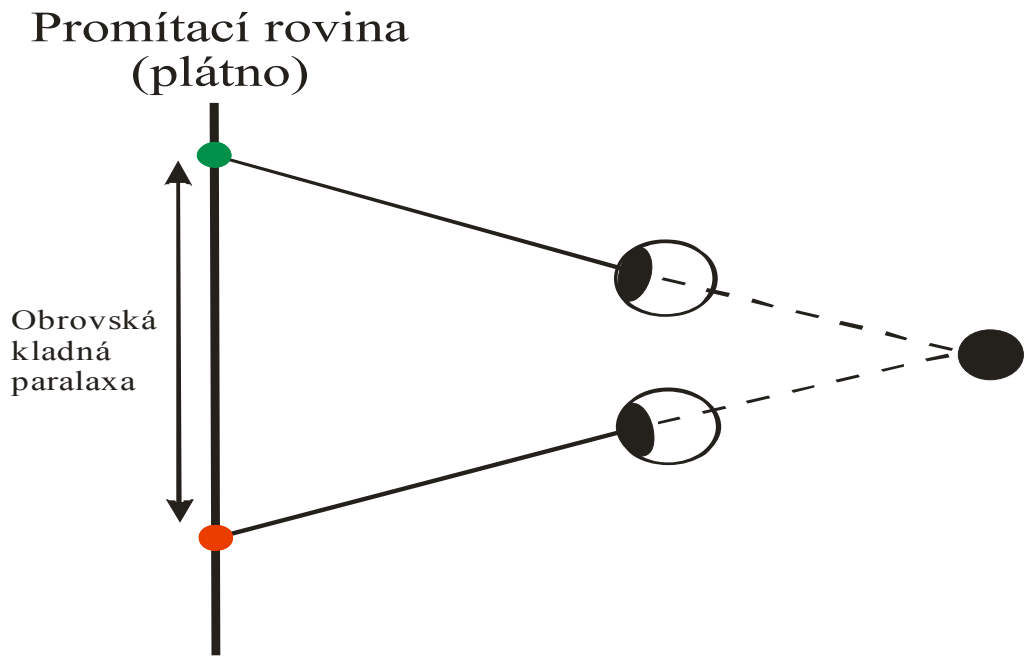
Ilustrace 2: záporná paralaxa



Ilustrace 3: ořezávání objektů

2.1.4 Kladná divergující paralaxa

Pokud by se v obraze objevila příliš velká kladná paralaxa (Ilustrace 4: kladná divergující paralaxa - ilustrace je záměrně přehnaná), oči by přestaly konvergovat a místo toho by divergovaly – osy očí směrem k plátnu by se rozbíhaly a bod konvergence by byl za námi. Je-li to možné, měli bychom se tomuto stavu vyhýbat, jelikož je velice namáhavý na oči a může způsobit bolest hlavy. V reálném světě se s tímto stavem nikdy nesetkáme.



Ilustrace 4: kladná divergující paralaxa

2.2 Další parametry

Kromě paralaxy je také mnoho jiných parametrů jenž ovlivňují vnímání hloubky v obraze. Nyní si řekneme něco o většině z nich. Většina z nich vychází ze zkušeností s reálným světem. Často jsou také nazývány jako monokulární zrakové podněty (monocular visual cues [3]) jelikož se uplatňují i při sledování obrazu z jediného bodu – jedním okem.

2.2.1 Pohybová paralaxa

Máme-li ve scéně dva a více pohybujících se objektů stejnou rychlostí, můžeme u objektu, který je k nám blíže pozorovat větší úhlovou rychlost. Z naších zkušeností plyne, že předmět, který se hýbe rychleji, je k nám blíže. Tento jev je ještě posílen, pokud se předměty pohybují po kruhové dráze se středem v bodě pozorovatele. Naopak pokud by se předměty pouze vzdalovaly a nevznikala by tak úhlová rychlost, tento jev se pravděpodobně vůbec neprojeví.

2.2.2 Světlo a stín

V reálném světě jsme zvyklí na jeden či více zdrojů světla a tudíž každá věc má svůj stín nebo dokonce více stínů. Stíny nám pomáhají orientovat se v prostoru. Díky stínu máme základní představu, kterým směrem je zdroj světla a jak daleko je asi předmět, na kterém vidíme stín, od předmětu, který stín vytváří.

2.2.3 Překrývající se objekty

Díváme-li se na dva předměty u kterých alespoň trochu známe tvar a jeden z nich zakrývá ten druhý, pak nám zkušenost říká, že zakrytý objekt je dále od nás než objekt zakrývající. Tohoto tématu jsme se již dotkli v kapitole 2.1.3 Záporná paralaxa.

2.2.4 Velikost

Dalším důležitým vodítkem pro náš mozek při vnímání hloubky je velikost objektu. Čím větší je objekt, tím blíže se nám jeví. Avšak i zde člověk vychází ze svých zkušeností a znalostí různých předmětů.

2.2.5 Barva a barevný odstín

Lidské oko každou barvu vnímá trochu jinak. Využijeme-li tohoto jevu tak, že objekty v popředí budou mít výraznější barvy (červená, oranžová, žlutá) než objekty v pozadí (např. modrá či zelená), můžeme vylepšit naši stereoskopickou projekci. Roli zde však nehrají pouze barvy, ale taky jejich odstíny způsobené atmosferickými vlivy (jako je například mlha, prach, smog, déšť atp.). Velmi vzdálené objekty se pak jeví více do modra.

2.2.6 Hustota stejných objektů

Je-li v jedné scéně mnoho stejných objektů, pak se také často předpokládá, že ve větší vzdálenosti budou menší a bude jich podstatně více než těsně u pozorovatele. Klasickým příkladem může být pohled z výšky na vinice či obyčejný les.

2.3 Zobrazování stereoskopických obrazů

Pro zobrazování stereoskopického snímku se v dřívějších dobách používal stereograf (často také nazýván stereoskop). Je to přístroj, jenž má dvě kukátka, přes které se díváme na dvě nezávislé stereofotografie, které pochopitelně musely být pořízeny současně. Díky optické soustavě stereoskopu každé oko vidí jen svůj obrázek a opět vzniká prostorový vjem. Stereofotografie zažívaly svůj největší úspěch v šedesátých letech minulého století, od té doby jsou však stále na ústupu.

Ilustrace 5: Stereoskop – převzat z [4]

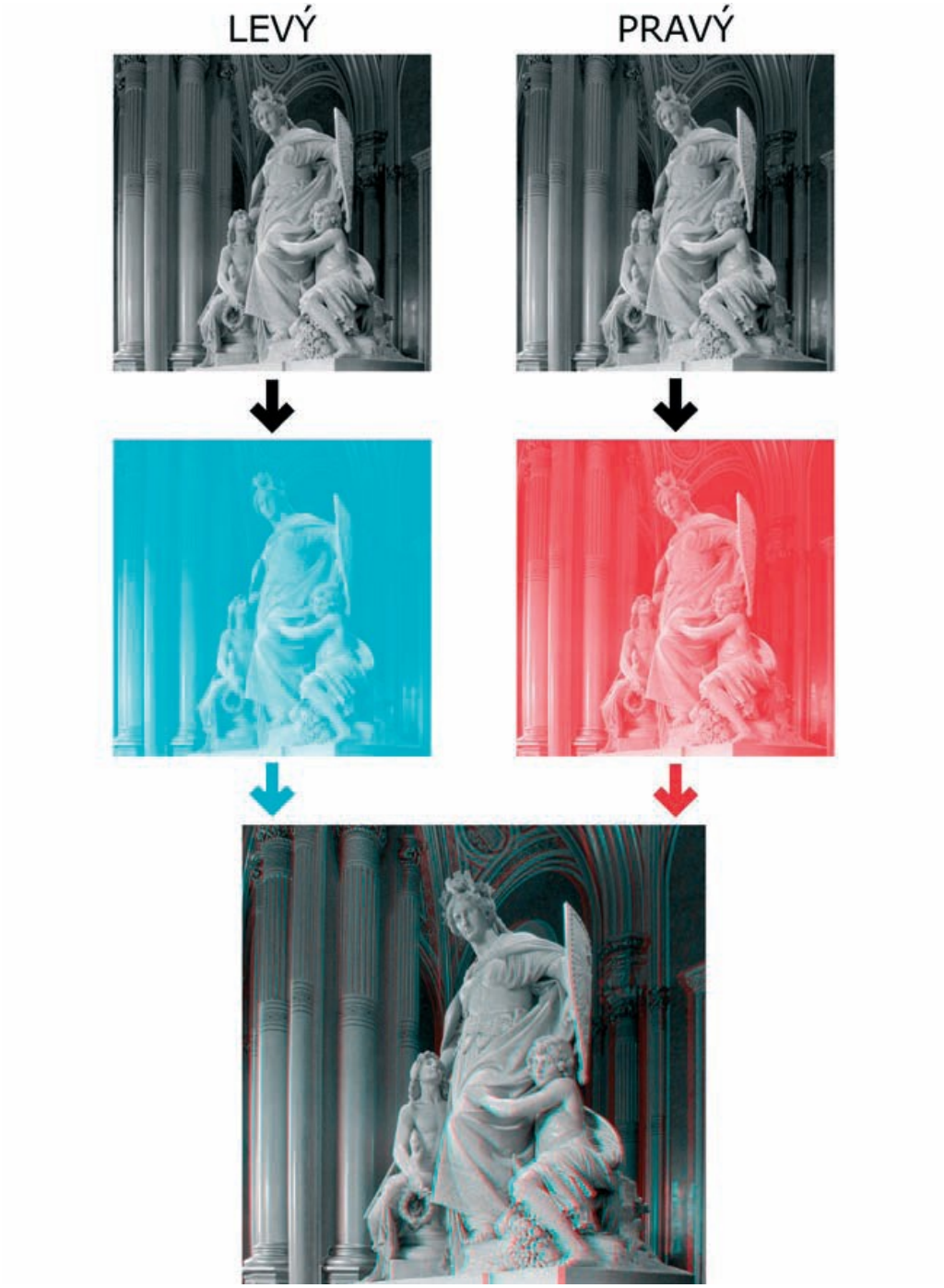
Myšlenka stereoskopu se však neztratila, byla pouze přetransformována dobou. Dnes jsou k dostání různé brýle či audiovizuální helmy pro vytvoření virtuální reality. Tyto helmy jsou realizovány pomocí malých LCD displejů, na kterých se promítá obraz a pomocné optiky. Tyto zařízení však vůbec nejsou levnou záležitostí. Ceny se mohou pohybovat i v mnoha desítkách tisíc korun. Další nevýhodou je, že se na daný obraz může dívat pouze jeden člověk v jednu chvíli. Pokud by se chtělo dívat více lidí, bylo by zapotřebí více těchto zařízení.

Další možný způsob je promítání obrazů na promítací plátno, kde se na to může pohodlně dívat mnoho lidí současně, všichni však musí mít speciální brýle. Nejběžnější využití je v různých kinech či zábavních parcích. Existuje však několik takovýchto technologií a můžeme je rozdělit na aktivní a pasivní podle brýlí jež využívají. V případě aktivní technologie nám stačí jediná promítačka, musí však být schopná produkovat dvojnásobný počet snímku za sekundu, to znamená 100, většinou však 120. Každý lichý snímek je pro jedno oko, sudý pro druhé oko. Jelikož se střídají co 10ms člověk si toho ani nevšimne. Pozorovatel pak musí mít brýle jež odfiltrují snímky tak, aby každé oko dostalo pouze svůj snímek. Toho je obvykle dosaženo pomocí zatmívacích LCD pro každé oko zvlášť. Brýle však musí být nějak synchronizovány s projektorem, ať už je to pomocí drátu, což není zrovna nejpohodlnější, ale brýle se nemusí nabíjet, anebo bezdrátově například infračerveným světlem. V takovém případě však musí mít baterku a po použití je nutno je znovu nabít.

Z tohoto pohledu má pasivní technologie výhodu, jelikož brýle jsou lehčí a nemusí se nabíjet. Jedna z pasivních technologií je založena na principu polarizovaného světla. Při tomto způsobu zobrazování jsou zapotřebí dva projektory. Před každý z nich se umístí polarizační fólie, ale otočená o 90°. Tím je zajištěno, že každý projektor bude produkovat správně polarizované světlo. Anebo projektor musí produkovat polarizované světlo jako přirozený výstup – např. LCD projektory. Každé sklo v brýlích má nanesenou polarizační fólii, opět jsou mezi sebou otočeny o 90°, tak aby propustily pouze správný obraz pro každé oko. Jednou z nevýhod této technologie je fakt, že nelze promítat na libovolné plátno. Plátno totiž musí být pokoveno (obvykle postříbřeno), nebo jinak upraveno, aby se zachovala polarizace odraženého světla. Tato technologie byla využita v této práci.

Další pasivní technologií je anaglyf – poprvé popsaný W. Rollmanem roku 1853 (jméno a letopočet převzato z [5]). Tato technika má sice velké nevýhody, ale díky své cenové dostupnosti a snadné výrobě přetrvala dodnes. Při využití anaglyfu se obraz pro levé a pravé oko filtruje pomocí barevných komplementárních fólií. Pokud bychom takovéto fólie položili na sebe, nemělo by přes ně být nic vidět, v praxi to však není moc dosažitelné. Anaglyfů lze udělat několik druhů, zřejmě nejběžnějším je červená fólie na levém oku, azurová na pravém (anglicky red-cyan). Daly by se také vytvořit také zeleno-purpurový, žluto-modrý a další, které by však již nebyly úplně komplementární a tedy i jejich vlastnosti by byly ještě horší. Hlavní nevýhodou je velká ztráta barevnosti, jelikož každým okem vidíme obraz v jiných barvách a výsledná barva se musí sestavit až v mozku. Další

nevýhodou je, že například čistě červený objekt nelze rozložit na dvě složky (obsahuje totiž pouze červenou) a tedy ho nelze zobrazit prostorově. Podobně to platí i pro jiné barvy. Dalším problémem je zobrazování objektů určitých barev na pozadí podobné barvy.



Ilustrace 6: tvorba anaglyfu, obrázek převzat z [6]

2.3.1 Ghosting

Při některých metodách zobrazování se nevyhneme takzvanému ghostingu (někdy nazýván také jako crosstalk). Tento jev se projevuje tím, že jedním okem vidíme i část obrazu, který je určen pro druhé oko. Nutno podotknout, že tento jev je velmi nežádoucí, jelikož rapidně snižuje kvalitu stereoskopického efektu a vždy se ho proto snažíme co nejvíce eliminovat. Při použití aktivní zatmívací technologie je důležité, aby LCD propouštěly světlo pouze v době, když je zobrazen snímek pro dané oko, pokud ne, nesmí propustit žádné světlo. Stejně tak je důležité, aby monitor či promítačka neměly žádný dosvit. U polarizační technologie se snažíme použít kvalitní polarizační filtry, dobře je seřídit jak na straně projektorů, tak také u brýlí. Divák by měl mít brýle nasazené rovně a hlavu také nenahýbat do stran. U obou těchto technologií to však není tak zlé jako u anaglyfu. Ten, jak již bylo řečeno, má mnoho špatných vlastností, ale tato se jevila při experimentování jako nejhorší. Ghostingem se podrobně zabývá p. Woods například v pracích [7,8].

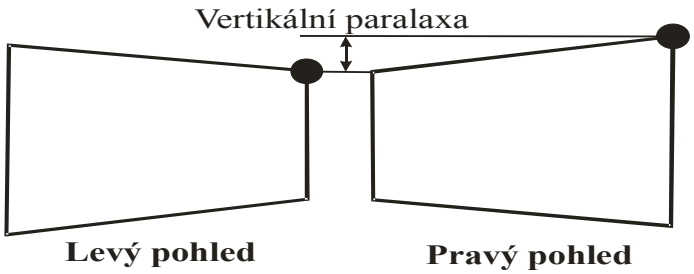
2.4 Tvorba stereoskopického obrazu

V této podkapitole si popíšeme jak vytvořit stereoskopický obraz a dále pak jak ho vytvořit aby nebyl moc náročný na oči. Existují dvě metody jak nastavit kamery a následně renderovat stereoskopický obraz. Metody jenž vytváří vertikální paralaxu můžeme označit jako nekorektní a měly by se používat jen v opodstatněných případech (například snímání obrazu reálnou příliš velkou kamerou či fotoaparátem, tak že by nebylo možné dát je těsně vedle sebe).

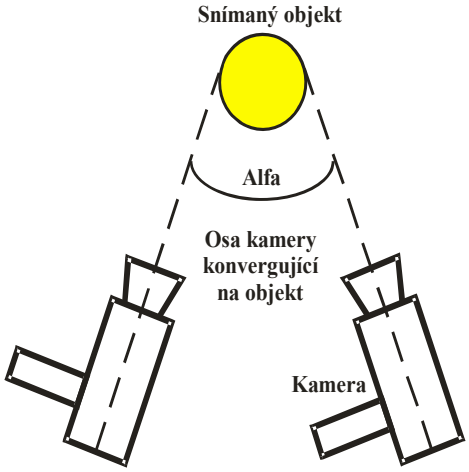
2.4.1 Toe-in

První metodou je Toe-in, při které osy kamer konvergují na snímaném objektu. Tato metoda je jednodušší na implementaci, jelikož nemusíme nic extra počítat, stačí nám nastavit směr každé kamery na náš objekt, o který máme největší zájem.

Při této metodě vzniká vertikální paralaxa na okrajích snímku. To může pozorovateli znepříjemnit sledování a dokonce přivodit bolest hlavy. Jediným vhodným případem je pokud snímáme poměrně malý objekt uprostřed záběru a pozadí je jednobarevné. V tomto případě také vznikne vertikální paralaxa, ale díky jednobarevnému pozadí se neprojeví.



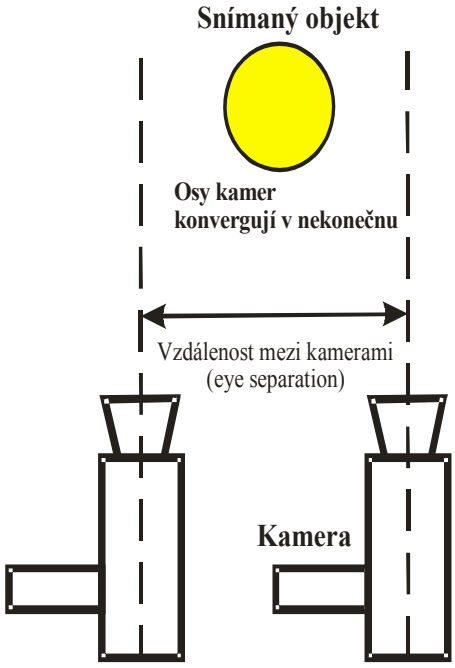
Ilustrace 8: Vznik vertikální paralaxy



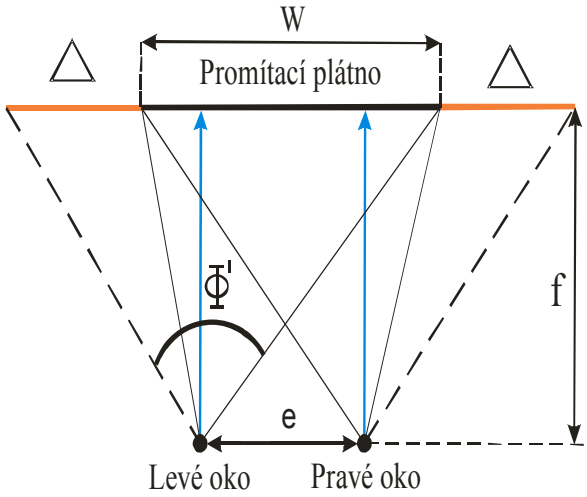
Ilustrace 7: Toe-in metoda (nevhodná)

2.4.2 Off-axis

Korektní metoda, při které nevzniká vertikální paralaxa [9]. Optické osy kamer konvergují v nekonečnu.



Ilustrace 9: Off-axis metoda



Ilustrace 10: Parametry stereoskopické projekce

Při vytváření takovéto scény musíme znát několik parametrů. Prvním z nich je vzdálenost, na kterou budeme ostřit (f), jinými slovy vzdálenost mezi kamerou a promítacím plátnem. Podle této

hodnoty určíme vzdálenost mezi kamerami (e) maximálně jako jednu dvacetinu. Pro menší namáhání očí se však doporučuje 1/30 nebo dokonce až 1/50. Čím menší vzdálenost mezi kamerami, tím méně náročné bude sledování obrazu, avšak relativní hloubka obrazu bude o něco menší (viz. Ilustrace 11: Relativní hloubka). Dalšími parametry je výsledná šířka (w) a výška (h) obrazu, který chceme vykreslit. Posledním parametrem je pozorovací úhel celé výsledné scény (Φ).

Dalším parametrem na Ilustraci 10 je Δ, což je počet pixelů, které se nezobrazí ve finálním obraze.

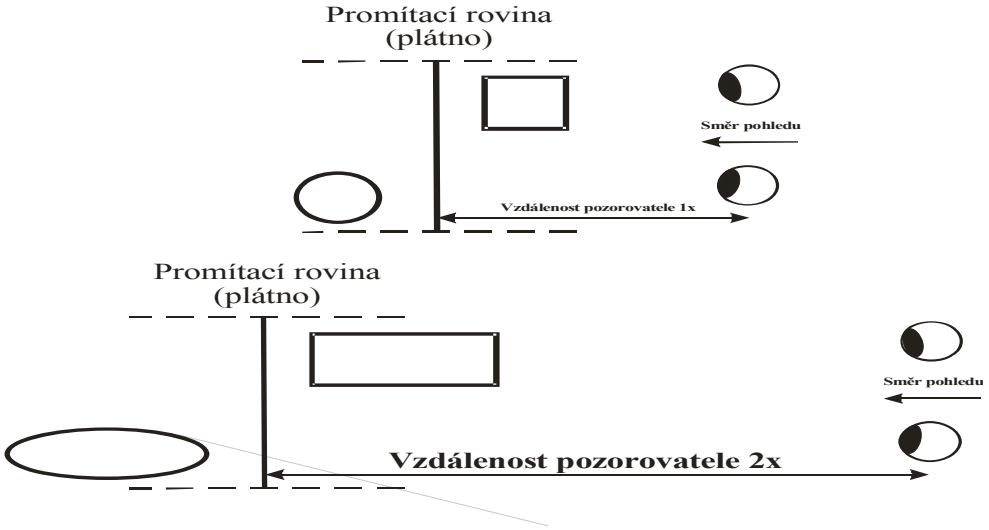
$$\Delta = \frac{w \cdot e}{2 \cdot f \cdot \tan(\Phi/2)}$$

Hodnoty e a f můžou být v libovolných jednotkách, jelikož se tyto jednotky navzájem vykrátí a zůstane jen poměr těchto hodnot. Hodnota w a Δ je obvykle v pixelech, možno však použít i jiné jednotky, vyžaduje-li to situace.

Posledním parametrem na Ilustraci 10 je Φ', což je pozorovací úhel jedné kamery před ořezáním na stereoskopickou projekci. Ten můžeme spočítat jako:

$$\Phi' = 2 \cdot \arctan \frac{(w + \Delta) \cdot \tan(\Phi/2)}{w}$$

Toto je teoretické podání, v kapitole o implementaci se na to podíváme z praktického hlediska.



Ilustrace 11: Relativní hloubka

3 2D & 3D grafika

V této kapitole si popíšeme něco o trojrozměrné grafice, jenž se přímo dotýká této práce. Aby aplikace, která vykresluje 3D grafiku, mohla běžet v reálném čase je zapotřebí veškeré vykreslování hardwarově akcelerovat. K tomuto účelu drtivá většina dnešních počítačů obsahuje grafický akcelerátor – ať už integrovaný na základní desce či jako samostatnou kartu připojenou přes sběrnici ke zbytku počítače.

Samotný grafický akcelerátor však není dostatečný k vykreslování námi požadovaného obsahu. Potřebujeme ještě určité softwarové rozhraní, přes které budeme přistupovat k možnostem, které poskytuje daný akcelerátor. Takovýchto knihoven existuje více, nejznámější je zřejmě DirectX a OpenGL. Nyní si uděláme krátké porovnání těchto dvou knihoven a dále se pak podíváme na další důležité knihovny z pohledu této práce.

3.1 DirectX

Hned na úvod je třeba říct, že DirectX bylo vyrobeno firmou Microsoft Corporation a je nadále touto firmou spravováno a aktualizováno. Aktuální verze (duben 2010) je DirectX11 [10]. DirectX není multiplatformní, to znamená, že ho lze využívat pouze na počítačích s instalovanými Windows. S ohledem na tržní zastoupení tohoto operačního systému to není zas až tak velké omezení. Různé verze Windows podporují různé verze DirectX.

DirectX zajišťuje standardizovaný přístup k hardwaru a v případě absence hardwarové podpory zajišťuje také emulaci všech funkcí této knihovny. Díky tomu se programátor nemusí starat o to, zda danou funkcionalitu hardware podporuje, ale může se zaměřit na svou vlastní aplikaci. DirectX není pouze rozhraním ke grafickému akcelerátoru, ale i ke zvuku a dalším hardwarovým či softwarovým komponentám počítače. DirectX se skládá z mnoha komponent. Ty hlavní si popíšeme trochu podrobněji:

- DirectDraw – využívá hardwarové akcelerace pro kreslení ve 2D, hlavními stavebními prvky jsou takzvané plochy – Surfaces, na které se umisťují objekty a následně vykreslují.
- Direct2D – ve Windows Vista a Windows 7 by mělo nahradit DirectDraw a přidává další funkcionalitu pro kreslení rastrové a vektorové grafiky.
- DirectWrite – slouží pro renderování textů a je dostupné až od Windows Vista a Windows 7.
- Direct3D Immediate Mode – slouží pro vykreslování 3D grafiky, pracuje na nejnižší vrstvě, tedy s grafickými primitivy. Z tohoto pohledu se podobá OpenGL.
- Direct3D Retained Mode – na rozdíl od Immediate Mode nepracuje s grafickými primitivy, ale rovnou s celými objekty. Taktéž využívá co možná nejvíce hardwarovou akceleraci.

- DirectSound – jedná se o komponentu DirectX, která se stará o přehrávání zvuků.
- DirectSound3D – rozšiřuje DirectSound o možnosti zpracovávat prostorové zvukové efekty.
- DirectMusic – je nadstavbou nad DirectSound umožnující lepší práci s hudbou, avšak počínaje Windows Vista je zavrhována.
- DirectInput – stará se o práci se vstupními zařízeními jako je myš, klávesnice, joystick a dalšími.
- DirectPlay – slouží pro snadnou síťovou komunikaci jak po lokální síti tak i po internetu.

3.2 OpenGL

Open Graphics Library, zkráceně OpenGL nebo někdy také OGL, jak již název napovídá, se na rozdíl od DirectX se specializuje pouze na vykreslování grafiky. Slouží tedy jako otevřená (oproti Direct3D, který je proprietární) definice pro aplikační rozhraní na grafický hardware. Oproti DirectX není nijak specifikován operační systém, na kterém může být využíváno, proto ho můžeme najít jak ve Windows tak i v Linux/Unixu nebo Mac OS nebo dokonce v různých vestavěných a mobilních zařízeních. Windows 98, Me a 2000 podporují OpenGL verzi 1.1.

Ilustrace 12: logo OpenGL

OpenGL je založeno na principu stavového automatu. To znamená, že pokud změníme nějakou stavovou hodnotu, zůstane v tomto stavu do té doby než ji jiná část programu jinak nenastaví. To občas může mít své výhody, ale i své nevýhody. Proto OpenGL nabízí zásobníkovou architektůru pro ukládání všech stavů. OpenGL na rozdíl od Direct3D Retained Mode nepodporuje práci s komplexními objekty a programátor si tedy musí vystačit pouze s grafickými primitivy jako je bod, úsečka, polygon. Pro urychlení práce se používají vertex buffer object, vertex arrays či display listy. Dále OpenGL nabízí různé způsoby texturování, osvětlení, transformace, řešení viditelností objektů, průhlednost a blending, antialiasing, práce s mlhou a mnohé další funkce. Navíc v OpenGL se často využívají různá rozšíření, které jsou podporovány jen některými výrobci grafických karet či jen některými kartami, čímž vzniká jistá nekompatibilita. Některé rozšíření jsou však v dnešní době běžné natolik, že je podporují snad všechny grafické karty. OpenGL nenabízí žádnou podporu pro vytvoření okna. K tomuto účelu je nutno použít jinou knihovnu (například SDL či GLUT) anebo okno vytvořit ve vlastní režii pomocí systémových funkcí. Stejně tak různé další podpůrné funkce jako například načítání textur nebo objektů ze souboru nejsou součástí a opět musíme použít jiné knihovny. V této práci byla využita právě knihovna OpenGL.

Při práci s OpenGL se často setkáme s knihovnami GLU, GLUT, freeglut a GLUI:

- GLU – OpenGL Utility Library je podpůrná knihovna často dodávaná přímo k OpenGL. Slouží pro zpříjemnění práce a umožňuje používat funkce na vyšší úrovni. Nabízí funkce jako například tesselaci, generování mipmapových textur, kreslení křivek, lepší interpretaci chybových kódů anebo zavádí nové primitiva. Všechny tyto funkce uvnitř volají klasické OpenGL rutiny. Ve skutečnosti tedy tato knihovna nepřidává novou funkcionalitu OpenGL, ale pouze usnadňuje práci s ním. Je multiplatformní a všechny funkce začínají prefixem `glu`.
- GLUT – OpenGL Utility Toolkit je také podpůrná knihovna, ale na rozdíl od GLU se zaměřuje na práci s operačním systémem. Tedy umožňuje jednoduše používat vstupně-výstupní operace, vytvořit okno, zachytávat události od myši a klávesnice a vytvořit jednoduchou rolovací nabídku. Navíc má několik funkcí pro vykreslení některých složitějších 3D objektů. Také je multiplatformní a všechny funkce mají prefix `glut`. Aktuálně se však již nevyvíjí.
- Freeglut je obdoba knihovny GLUT, obsahuje stejně pojmenované funkce, ale navíc k tomu přidává nové, které ošetřují chyby nebo nedokonalosti v původní knihovně. Tato knihovna by měla sloužit jako náhrada ke stagnující knihovně GLUT. Je šířena pod licencí X-Consortium [11].
- GLUI – OpenGL User Interface Library je nadstavba nad knihovnou GLUT pro jazyk C++, pomocí které se dají snadno vytvářet grafické ovládací prvky jako jsou tlačítka, check-boxy, radio-buttony a podobné. Jelikož je postavená nad knihovnou GLUT je také multiplatformní. Všechny funkce a datové typy začínají prefixem `GLUI_`.

3.3 SDL

Simple DirectMedia Layer, zkráceně SDL, je multiplatformní open-source knihovna pro jazyk C, jež nabízí možnosti vykreslování 2D grafiky, přehrávání zvuků, obsluhu vstupních zařízení a další komunikaci s operačním systémem. Jak napovídá název knihovny, jedná se v podstatě o vrstvu mezi operačními systémem a uživatelskou aplikací. Tato knihovna je šířena pod dvěmi licencemi. SDL verze 1.2 má pouze licenci GNU LGPL, verze 1.3 má navíc i komerční licenci, která umožňuje statické linkování v komerčních produktech na platformách, které nepodporují dynamické linkování (např: Nintendo DS). SDL je rozděleno do několika knihoven, které se podle potřeby přilinkují k výslednému programu. Hlavní modul má na starosti vytvoření okna, vykreslování grafiky, správu událostí, základní možnosti přehrávání zvuků, práci s časem, obsluhu CD-ROM a další. SDL také nabízí podporu pro vícevláknové programování. Funkce a datové typy mají prefix `SDL_`. Dalšími moduly jsou:

- SDL_sound – další možnosti se zvukem, hlavně tedy dekódování zvuků z běžně rozšířených formátů jako je MP3, MID, OGG. Funkce a datové typy mají prefix `Sound_`.
- SDL_mixer – další knihovna zabývající se zvuky, tentokrát se ale specializuje na přehrávání a mixování hudby a další audio efekty. Funkce a datové typy mají prefix `Mix_`.
- SDL_ttf – knihovna jenž slouží pro základní výpis textů s ohledem na použitý TrueType font. Funkce a datové typy mají prefix `TTF_`.
- SDL_rtf – slouží pro jednoduché zobrazování souborů ve formátu RTF[1]. Funkce a datové typy mají prefix `RTF_`.
- SDL_net – jednoduché rozhraní pro síťovou komunikaci přes TCP a UDP. Funkce a datové typy mají prefix `SDLNet_`.
- SDL_image – poslední a zřejmě nejdůležitější oficiálně podporovaný modul. Slouží pro načítání obrázků ze souborů různých formátů včetně PNG, JPG, GIF, TGA a dalších. Funkce mají prefix `IMG_`.

Dále existuje obrovské množství knihoven (viz. oficiální WWW stránky SDL [12]), které nějakým způsobem vylepšují práci s klasickým SDL API, ať už se jedná o nadstavby pro C++ či nejrůznější ovládací prvky. Pro SDL je také velké množství wrapperů pro další jazyky kromě klasického C. Jelikož se SDL dá snadno propojit s OpenGL, bylo využito v této práci.

3.4 Lib3ds

Lib3ds je také multiplatformní knihovna (oficiální WWW stránky [13]), která slouží pro spravování 3D-Studio Release 3 a 4 (3DS) souborů. Je další alternativou k oficiálnímu Autodesk's 3DS File Toolkitu anebo knihovně scene3ds, které obě slouží pro načítání a ukládání z/do 3DS souborů. Tato knihovna byla původně pod licencí GNU GPL, ale v roce 2000 byla kompletně přepsána do ANSI-C pro lepší hardwarovou nezávislost. Aby mohla být použita pro komerční využití, její licence se také změnila na GNU LGPL. V současné době jsou ke stažení dvě verze knihovny. První 1.3.0 naposledy upravena 28.6.2007 a verze 2.0.0 RC1, která se aktuálně stále vyvíjí. Knihovna podporuje:

- ukládání a načítání
  - objektů
  - materiálů
  - kamer
  - světel
  - hierarchie
  - klíčových snímků

1 RTF – Rich Text Format

    - atmosferického nastavení
    - nastavení pozadí
    - nastavení stínových map
    - nastavení wievportu
- vyhodnocení všech animačních klíčových snímků
- jednoduchou práci s datovými typy
    - vektor
    - quaternion
    - matice
- snadné začlenění k OpenGL

3.5 Qt

Qt je jedna z nejpopulárnějších knihoven pro tvorbu grafického uživatelského rozhraní. Je napsána v jazyce C++ a je založena na dědičnosti. Při použití této knihovny v jazyce C++ doplňuje jazyk o vlastní klíčová slova `signal`, `slot` a `emit`. Tyto klíčová slova usnadňují práci při propojování objektů, tak aby mohly mezi sebou snadno komunikovat – zasílat si zprávy. Zavádí tedy vlastní Meta-Object System. Tato skutečnost však komplikuje překlad, jelikož je nutno před vlastním C++ překladem ještě použít speciální preprocesor MOC[2]. Každá námi vytvořená třída, ve které chceme využívat signálů a slotů, musí být odvozena z třídy `QObject` (nebo třídy odvozené z `QObject`). Navíc v privátní sekci musí mít makro `Q_OBJECT`. Kromě signálů a slotů zavádí Meta-Object System také podporu pro RTTI[3] a DPS[4].

Jelikož je multiplatformní lze programy snadno přeložit ve Windows, Windows CE, Linuxu i na Mac OS X. Je to skutečně obrovská knihovna, obsahuje více než 1000 tříd. Aktuální verze je 4.6.2, tato verze podporuje i tvorbu programů bez grafického rozhraní. Kromě podpory pro GUI má podporu i pro další věci jako například soubory, síť, procesy, vlákna, databáze či XML. Qt má vlastní paměťový management.

Součásti distribuce je i Qt Creator a Qt Designer pro usnadnění tvorby grafického rozhraní. Dále pak QMake pro usnadnění překladu a pochopitelně výše popsaný preprocesor MOC.

2 MOC – Meta-Object Compiler

3 RTTI – Run Time Type Information

4 DPS – Dynamic Property System

4 ROBE Lighting s.r.o.

ROBE je česká firma se sídlem v Hážovicích, výrobními prostředky ve Valašském Meziříčí a obchodním zastoupením po celém světě. Firma se zabývá především zábavní a osvětlovací technikou, která nachází uplatnění jak na obyčejných zábavních akcích a diskotékách, v nejrůznějších televizních pořadech či koncertních síních a divadlech, tak i pro nasvětlování budov a staveb po celém světě. Jejich sortiment se skládá z 15 kategorií, dohromady přes 100 druhů výrobků. Všechna světla je možné ovládat pomocí DMX protokolu.

Historie firmy sahá do roku 1993, kdy Ladislav Petřek a Josef Valchář začali projektovat a vyrábět svá první světla. Z počátku na trhu vystupovali pod názvy jiných firem – Movietec (Německo), Starway (Francie), Sagitter (Itálie), postupem času však začali vyrábět pod vlastní značkou. První veletrh, na kterém se prezentovaly jejich světla, byl Frankfurtský *ProLight & Sound* 1994. Od té doby také každoročně prezentují své nové výrobky na veletrhu v Rimini a Londýně. Dnes má firma přes 330 přímých zaměstnanců a výrobní prostory 20.000 čtverečních metrů. Vyrobená světla se prodávají ve vice než 90 zemích světa. Firma také založila dceřinou společnost ANOLIS, která se specializuje na výrobu LED světel.

Ilustrace 13: výrobní hala ve Valašském Meziříčí

4.1 Serie DigitalSpot

DigitalSpot je poměrně nový typ světel, který byl poprvé prezentován na světové roadshow *Future is Digital*. Od té doby se stále vyvíjí a nabízí stále nové možnosti. Původně se vyráběly 3 verze, dnes se od prostředně výkonné již upustilo a vyrábí se jen verze DigitalSpot 7000 a 3000. Tento typ světel je

ideální pro profesionální zábavní průmysl včetně koncertních turné, TV show a divadla. Hlavní rozdíly jsou ve světelném výkonu a tím pádem i ve spotřebě, velikostí pevného disku, počtu digitálních gobo vrstev, výkonu dosvětlovacích LEDWash modulů a typu projekční technologie. Jelikož DigitalSpot 7000 vyzařuje již polarizované světlo, nehodí se pro zobrazování stereoskopických obrazů pomocí polarizační technologie. Proto se v této práci využívalo výhradně DigitalSpotu 3000.

DigitalSpoty mají v sobě vestavěný klasický počítač, na kterém běží operační systém Linux. Základním grafickým prvkem je zde vrstva, na kterou se může promítat libovolný obraz, video či efekt. Tyto vrstvy (slabší verze světla má 3, výkonnější 4) jsou umístěny nad sebou a každá vrstva může buďto zpracovávat předchozí vrstvu anebo vykreslovat vlastní prvky a tyto vrstvy se pak navzájem spojí nějakou matematickou operaci (kopie, plus, minus, krát, maximum, minimum). Co se týče obrázků a videí, přímo od výroby je zde velký výběr, ale i uživatel si může dále doplnit vlastní, omezením je pouze velikost pevného disku, které je u DigitalSpotu 7000 500GB, zatímco u DigitalSpotu 3000 jen 160GB. Různých grafických efektů je více než 160. Výsledné množství po kombinaci všech vrstev je nepřeberné.

Ilustrace 14: DigitalSpot 3000

5 Návrh

Doposud jsme stále popisovali věci, které se sice týkají této práce, ale pouze z teoretického hlediska. Nyní se tedy zaměříme na to samé, ale z praktického hlediska. Nejprve si popíšeme několik etap návrhu softwaru a poté přejdeme k implementaci.

5.1 Výběr knihoven

V první fázi vývoje bylo nutno rozhodnout jaké knihovny použít a hlavně poté nastudovat jejich možnosti. Použití OpenGL bylo jasné, to vyplývalo ze zadání.

5.1.1 Vytvoření okna a správa událostí

Dále bylo třeba rozhodnout zda použít SDL, GLUT nebo systémové API pro vytvoření okna a odchytávání událostí v testovacím programu. Z organizačních důvodů nebylo možné testovat přímo na DigitalSpotu. Systémové API bylo zavřeno hned ze začátku, jelikož by bylo zapotřebí udělat minimálně verzi pro Windows a verzi pro Linux. Na výběr tedy zůstalo SDL a GLUT. Ve většině studijních materiálech je využito GLUT, jelikož ale ve firmě ROBE používají SDL, bylo rozhodnuto také pro tuto knihovnu, aby se práce více přiblížila jejich praxi. Účel by splnila kterákoli výše popsaná možnost, jelikož testovací aplikace měla sloužit pouze pro účely této bakalářské práce a mé vlastní potřeby při testování.

5.1.2 Načítání objektů

Další rozhodování se týkalo načítání dat 3D modelů. Při konzultaci mi byla doporučena knihovna lib3ds pro načítání souborů s příponou .3DS. Knihovna lib3ds existuje ve dvou verzích (viz. Kapitola 2D & 3D grafika), tyto dvě verze se však od sebe docela zásadně liší. Jednak některé funkce pro stejnou činnost se v každé verzi jmenují jinak. Ale hlavně každá verze vytváří jinou strukturu dat v paměti a tedy použití jedné či druhé verze se zásadně liší. V této práci byla použita novější verze knihovny.

5.1.3 Tvorba úvodního menu

Poslední zásadní rozhodování se týkalo knihovny pro tvorbu grafického uživatelského rozhraní pro usnadnění testování aplikace. Jelikož toto rozhraní slouží pouze pro testování, jediný požadavek byl aby to bylo možné přeložit jak ve Windows tak i v Linuxu. Multiplatformních knihoven existuje

několik, byla zvolena knihovna Qt z důvodu poměrně snadné přenositelnosti a podpoře mnoha platforem.

5.2 Objektový návrh

Při výběru jazyka implementace bylo na výběr mezi C a C++, jelikož ale ve firmě ROBE využívají jazyk C++, stal se také implementačním jazykem této práce. Z tohoto předpokladu také vychází objektový návrh aplikace. Při návrhu byl kladen důraz na efektivitu a třívrstvou architekturu, kde nejvyšší vrstvou je vrstva *prezenční*, uprostřed je *business* vrstva a nejníže je *datová*. Datová vrstva se stará o práci s daty uloženými na pevném disku. V tomto případě se jedná o načítání 3D objektů. V business vrstvě je implementována logika aplikace včetně většiny práce s knihovnou OpenGL. Prezenční vrstva má na starosti vykreslení okna, do kterého se v business vrstvě vykresluje obsah. Dále pak se stará o komunikaci s uživatelem prostřednictvím operačního systému. A nakonec vypisuje aktuální parametry. Každá vrstva by měla ke své činnosti využívat pouze vrstvu těsně pod sebou a nesmí používat jakoukoli vyšší vrstvu. Business vrstva tedy může využívat pouze vlastních zdrojů a zdrojů z datové vrstvy, nikoliv však z prezenční.

5.2.1 Abstraktní třídy

Tato fáze návrhu se týkala již samotného softwaru. Při návrhu tříd se vycházelo z klasického objektového návrhu, kde jedna či více ABC[5] tvoří rozhraní pro třídy z nich odvozené. V tomto případě se jedná o třídu `StereoscopicEffect`. Tato třída v sobě kombinuje práci s kamerou a časem. Dále pak obsahuje předpokládanou velikost výsledného obrazu (v pixelech), pozorovací úhel a rozhraní pro vykreslování a předávání parametrů. Od této třídy jsou následně odvozeny třídy všech efektů. Ačkoli efekt zřejmě není nejlepší název, jako lepší se jeví například scéna, byl použit, jelikož ve firmě ROBE ho také používají. Po konzultaci s p. Farníkem byla tato třída odvozena od jeho abstraktní bázové třídy `GraphicsEffect`, jen lehce upravena a docílilo se tak sjednocení rozhraní. `GraphicsEffect` obsahuje jednu metodu pro předávání parametrů, tato třída přidává další jelikož původní 3 parametry se později ukázaly jako nedostatečné. Kromě této ABC byla navržena abstraktní třída `Obj_3D` pro načítání a zobrazování objektů. Od této třídy je následně odvozena třída Obj_3DS, která slouží pro načítání objektů z formátu 3DS a využívá výše popsanou knihovnu lib3ds. Původně bylo v plánu odvodit více takovýchto tříd pro načítání z různých formátů, nakonec k tomu však nedošlo.

5 Abstract Base Class – abstraktní bázová třída

5.2.2 Odvozené třídy

Od třídy `StereoscopicEffect` jsou odvozeny třídy efektů. Ty jsou `StarEffect`, `ButterflyEffect`, `AtomEffect`, `TextEffect` a `FountainEffect`. Jedná se o business vrstvu a nyní si je trochu přiblížíme. `StarEffect` je třída, která se stará o vykreslování scény, ve které pozorovatel proléta nekonečnou hvězdokupou. Tuto scénu je možné ovládat pomocí 5 parametrů. Jednak lze měnit počet hvězd, jejich rychlost a velikost, ale také způsob jakým se aplikuje předchozí vrstva a jaký tvar mají hvězdy mít. Ve scéně `ButterflyEffect` je vykresleno několik motýlů, kteří se jeví býti před promítacím plátnem. Všichni motýli jsou částečně průhlední a je možné volit počet motýlů, jejich typ a relativní hloubku motýlů. Dalším efektem je `AtomEffect`, ve kterém se zobrazuje jádro atomu s obíhajícími elektrony. Jádro je umístěno přesně na promítací plátno a elektrony se z plátna vynořují a opět zanořují za plátno. Je možné nastavovat několik způsobů rotace a 14 druhů aplikace předchozí vrstvy. Třída `TextEffect` slouží pro vykreslování prostorového textu, který se postupně vzdaluje od pozorovatele podobně jako ve Star Wars. Poslední scénou je `FountainEffect`, kde je vykreslena fontána s tryskající vodou. Je zde možné nastavit gravitaci, sílu větru a rotaci fontány.

5.2.3 Testovací aplikace

Testovací aplikace se skládá ze dvou částí. První částí je aplikace, ve které se vykresluje výsledný obsah, tato část je obsažena ve třídě `TstApp`. Testovací aplikace používá knihovnu SDL. Stará se o inicializaci, vytvoření okna a komunikaci s uživatelem a OS. `TstApp` spadá do prezenční vrstvy návrhu. Testovací aplikace se ovládá pomocí několika málo kláves, viz následující tabulka.

Q	Scéna s průletem hvězdokupou
W	Scéna s motýly
E	Scéna s rotujícím atomem
R	Scéna s plovoucím textem StarWars
T	Scéna s fontánou
1,2,3,4,5	Přepínání mezi parametry 1,2,3,4,5
A	Dlouhým stiskem pozvolné zvedání aktuálního parametru
Z	Dlouhým stiskem pozvolné snižování aktuálního parametru
Šipka ←	Přepnutí na pohled levým okem
Šipka →	Přepnutí na pohled pravým okem
Šipka ↓	Přepnutí na anaglyf
S	Přepínání mezi texturama
X	Přepínání mezi texturama

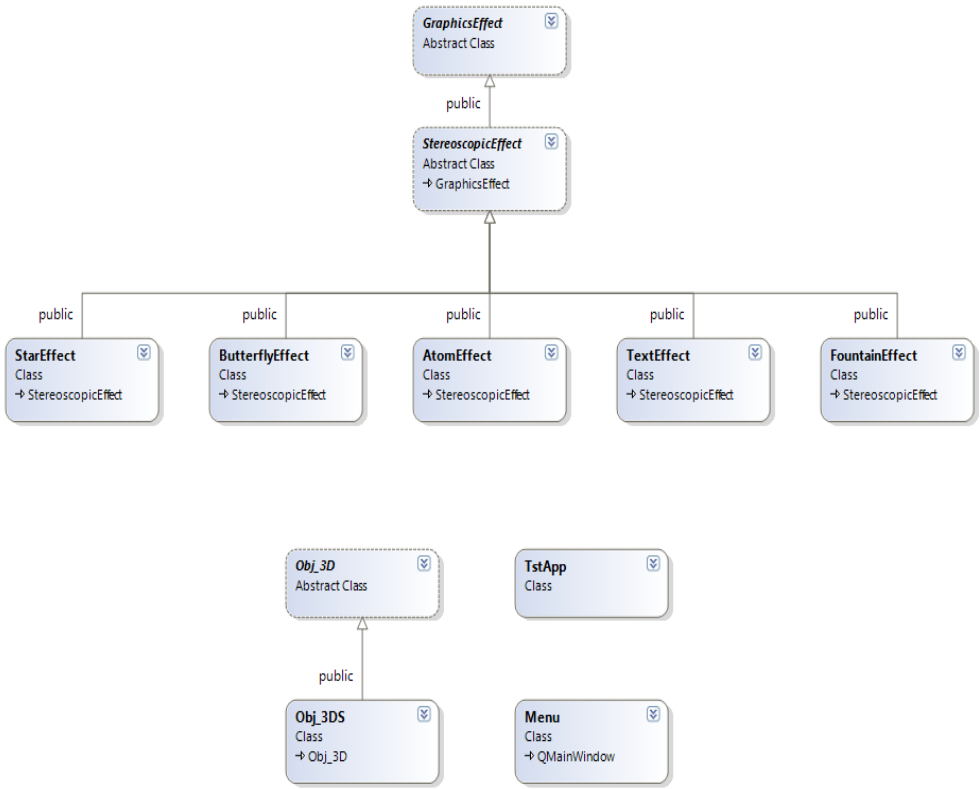
Po kompilaci se všechny doposud popsané třídy přeloží jako aplikace bin.exe spustitelná z příkazové řádky.

Druhou části testovací aplikace je grafické menu vytvořené pomocí knihovny Qt. Toto menu slouží hlavně pro zjednodušení a zpříjemnění testování. Menu je rozděleno na dvě části. Vpravo je Nápověda k programu, vlevo pak Počáteční nastavení, kde je možné volit výše popsané možnosti. Menu je popsáno třídou `Menu` a překládá se jako samostatný exe soubor, který vnitřně spouští bin.exe.

Poslední věc o které je třeba se zmínit je předávání simulační textury z testovací aplikace do jednotlivých scén. Tato drobnost nemohla býti navržena, ale bylo zapotřebí se přizpůsobit firmě ROBE, jelikož oni v každém efektu předpokládají že výsledek z předchozí vrstvy je aktivní v OpenGL a pokaždé se jen otestuje pravdivost tohoto předpokladu pomocí funkce `glIsEnabled(GL_TEXTURE_2D)`.

5.2.4 Navržená hierarchie tříd

Po dokončení návrhu je možné podívat se na jednoduchý diagram tříd. V příloze je pak celý včetně tříd vygenerovaných pomocí Qt designeru, všech struktur, typedefů a výčtových typů.



Ilustrace 15: Jednoduchý diagram tříd

6 Implementace

V této kapitole se podrobně podíváme na implementační detaily této práce. První věc, na kterou se zaměříme je testovací aplikace. Posléze přejdeme k popisu jednotlivých tříd a nakonec se podíváme na pomocné rutiny.

6.1 Třída TstApp

Testovací aplikace je rozdělena do tří souborů. Prvním je main.cpp, ve kterém je pouze vstupní bod programu, instanciace `TstApp` a spuštění nekonečné smyčky `Main_loop()`. Soubor TstApp.h obsahuje definici třídy `TstApp`. Třetím souborem je TstApp.cpp, ve kterém je implementace všech metod. Na začátku v konstruktoru proběhne inicializace knihoven SDL a OpenGL, vytvoření okna aplikace a zpracování vstupních parametrů. Hlavní nekonečná smyčka programu obsahuje volání metody `ZpracEvent(GLuint dt)` pro zpracování událostí, nastaví parametry pro aktuální efekt, aktivuje simulační texturu pokud nějaká byla vybrána, vykreslí scénu, vykreslí popis aktivních parametrů a nakonec pokud vše trvalo méně než 20ms, přenechá zbylý procesorový čas ostatním aplikacím. Metoda pro zpracování událostí obsahuje ošetření kláves (viz 5.2.3 Testovací aplikace), případně umožňuje změnu velikosti okna z původního rozměru 1024 x 768 na jiný.

6.2 Třída Menu

Tato třída obsahuje funkcionalitu pro grafické menu. Menu bylo vytvořeno pomocí programu Qt designer, který svůj výsledek uložil do souboru menu.ui. Při překladu se z tohoto souboru vygeneruje ui_menu.h, který obsahuje zdrojový kód v C++, konkrétně třídy `Ui_MainWindow` a `MainWindow`. Třída `Menu` tedy zapouzdřuje tyto třídy a přidává sloty, které reagují na změnu výběru scény a stisk tlačítka Start. Při stisku tlačítka se vytvoří nový `QProcess`, který se následně spustí s parametrem bin.exe a devíti parametry popisujícími nastavení menu. Při změně rolovací nabídky scén se překreslí popisy jednotlivých posuvníků a případně se některé skryjí nebo zobrazí.

6.3 Třídy Obj_3D a Obj_3DS

Abstraktní třída Obj_3D obsahuje kromě čistě virtuálních metod pro aktualizaci a vykreslování objektu také několik metod a atributů pro nastavení velikosti a rotace objektu. Na této třídě staví odvozená třída Obj_3DS pojmenovaná podle 3DS souborů se kterými pracuje. Hlavní roli zde hraje

konstruktor, který volá dvě privátní metody pro vytvoření display-listu. Pokud se v konstruktoru nepodaří otevřít a načíst požadovaný soubor pomocí funkce `lib3ds_file_open(const char*)`, která je z knihovny lib3ds, volá se první metoda, která jen nouzově vytvoří kostku, jinak se volá druhá metoda. Každý 3D model uložený v souboru .3DS se skládá z jedné nebo více sítí[6]. Nejprve se projdou všechny tyto sítě a spočítá se celkový počet plošek[7]. Poté podle počtu plošek můžeme alokovat paměť, do které budeme ukládat souřadnice bodů v prostoru[8], normálové vektory každého bodu a případně i texturovací souřadnice. Jakmile je alokována paměť, můžeme do ní překopírovat všechny data z paměťové struktury vytvořené knihovnou lib3ds. Poté se celý objekt normalizuje, jelikož modely z různých editorů nebo od různých autorů můžou mít různé velikosti. Nakonec se v jediném cyklu vytvoří požadovaný display-list a uvolní se veškerá použitá paměť. Vykreslovací metoda pouze nastaví velikost, pootočení a případně povolí texturování a poté zavolá vykreslení display-listu.

6.4 Abstraktní třídy GraphicsEffect a StereoscopicEffect

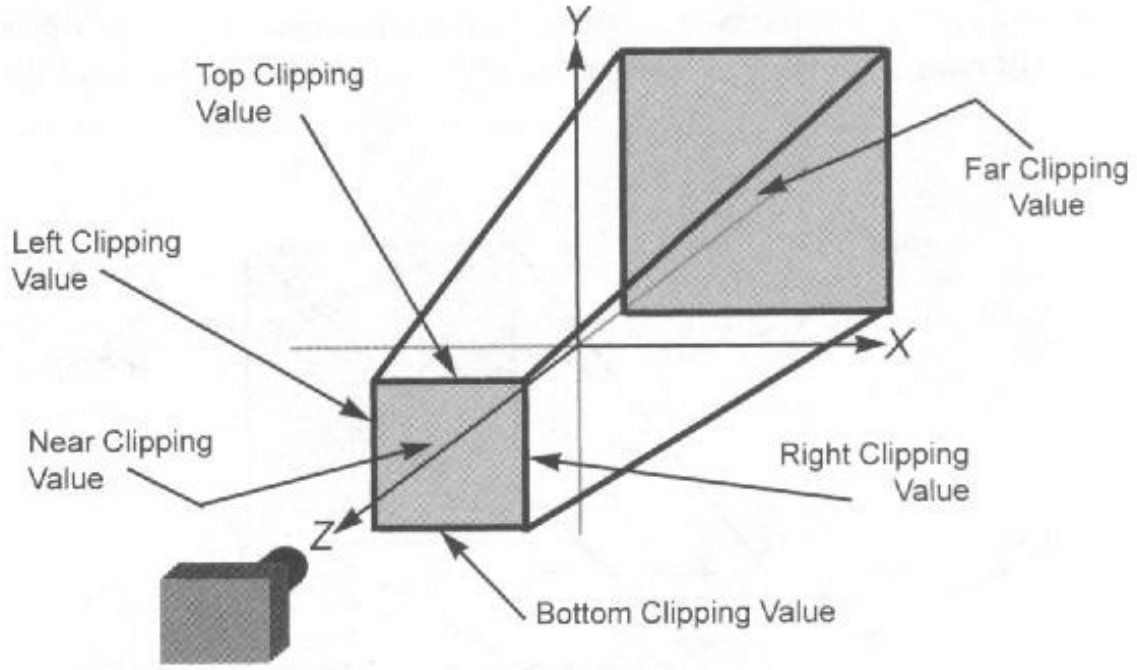
Celá třída `GraphicsEffect` byla převzata od p. Farníka z ROBE a tudíž do ní nebylo možné jakkoli zasahovat. Obsahuje základní rozhraní pro nastavení parametrů, synchronizaci, reinicializaci pohybu, barvy, rotace, zvětšení, posunu a vykreslování. V odvozených třídách je implementováno pouze rozhraní pro nastavení parametrů a vykreslování.

Třída `StereoscopicEffect` přidává nastavení kamery zda se jedná o pravé oko, levé oko či anaglyf. Dále pak několik metod pro posun kamery a původně byly zamýšleny i různé rotace kamery. Nakonec však rotací ani posunů nebylo zapotřebí jelikož ve všech scénách je kamera statická. Hlavním účelem této třídy zůstalo nastavení stereoskopické projekce. To provádí metoda `void setStereoProjection(bool leva, bool anaglyph, GLfloat focus)`. Tato metoda je založená na OpenGL funkci `void glFrustum( GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)`, která slouží pro vytvoření perspektivní projekce. Jednotlivé parametry jsou zobrazeny na Ilustraci 16. Na tomto obrázku nás v podstatě bude zajímat pouze pravoúhlý trojúhelník mezi kamerou, bodem kde se protíná Z-ová osa s přední ořezávací rovinou (nazvěme ho pomyslným středem) a okrajem přední ořezávací roviny. Trojúhelník znázorněn na Ilustraci 17.

6 Tzv. Mesh – síť bodů, které jsou navzájem provázány

7 Tzv. Face, každý face je trojúhelníkového tvaru a je ohraničen třemi body

8 Tzv. Vertex – množné číslo vertices, přeloženo z angličtiny znamená vrchol



Ilustrace 16: znázornění perspektivní projekce, obrázek převzat z [14]

V tomto trojúhelníku známe úhel α, který je polovinou pozorovacího úhlu a vzdálenost N což je vzdálenost mezi kamerou a přední ořezávací rovinou (Near clipping plane). Parametr X je vzdálenost mezi pomyslným středem a okrajem ať už se jedná o top, left, bottom nebo right. Následující vztahy nám udávají jak tyto parametry vypočítat pro levé oko. Parametr left:

$$-\frac{w}{h}\cdot N\cdot\tan(\alpha)+\frac{v\cdot 0{,}5\cdot N}{f}$$

parametr right:

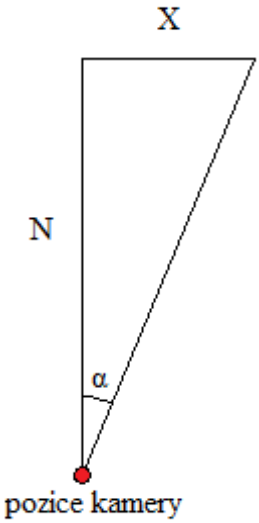
$$\frac{w}{h}\cdot N\cdot\tan(\alpha)+\frac{v\cdot 0{,}5\cdot N}{f}$$

parametr top:

$$N\cdot\tan(\alpha)$$

parametr bottom:

$$-N\cdot\tan(\alpha)$$



Ilustrace 17: znázornění počítaných hodnot pro funkci glFrustum

Pro pravé oko se vzorce liší jen trochu a pouze pro parametr left:

$$-\frac{w}{h}\cdot N\cdot\tan(\alpha)-\frac{v\cdot 0{,}5\cdot N}{f}$$

a parametr right:

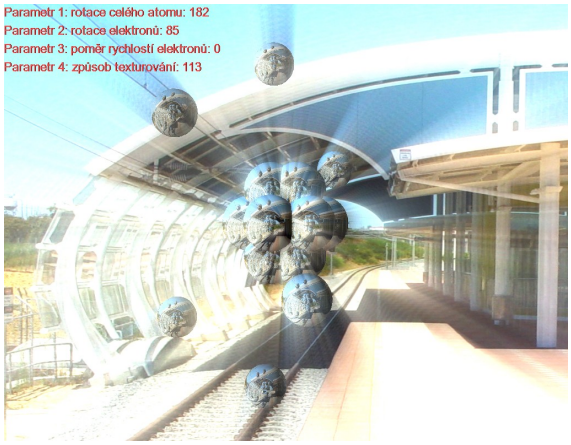
$$\frac{w}{h}\cdot N\cdot\tan(\alpha)-\frac{v\cdot 0{,}5\cdot N}{f}$$

V těchto vztazích je poměr *w*/*h* poměr stran obrazu, v základním stavu tedy 4/3. *N* je vzdálenost přední ořezávací roviny, *α* je polovina pozorovacího úhlu, *v* je vzdálenost mezi levým a pravým okem a *f* je vzdálenost, na kterou je zaostřeno. Stereoskopického jevu se dá dosáhnout i bez těchto vzorců, stačilo by nechat klasické symetrické ořezávání, vyrenderovat obraz ze dvou bodů a výsledky posunout tak, aby objekt na který ostříme měl nulovou paralaxu. Tato metoda má však velkou nevýhodu v tom, že na krajích obrazu jsou pruhy, kde je obraz jen z jednoho pohledu.

Další věc, kterou řeší metoda `setStereoProjection()` je převod obrazu do anaglyfu. K tomu se používá OpenGL funkce `glColorMask()`, které se předají 4 boolean parametry, jenž říkají, které barvy se mají vykreslovat a které ne.

6.5 Třída AtomEffect

Co se týče veřejných metod, tato třída implementuje dříve specifikované rozhraní v třídě `StereoscopicEffect`. V konstruktoru se nejprve inicializuje 14 částí jádra atomu a 6 obíhajících elektronů. K tomuto účelu slouží pomocná struktura `SubAtom`, která uchovává popis jediné částice. Poté se vytvoří dvě prázdné textury, do kterých se bude ukládat vyrenderovaný snímek a snímek pro rozmazávací efekt. Veškerá práce se odvíjí od metody `draw()`, nejprve se uloží všechny OpenGL atributy na atributový zásobník a na konci metody se opět obnoví. Je to důležité aby se nemohla ovlivnit jiná část programu jak již bylo popsáno v kapitole 3.2 OpenGL. Poté se nastaví všechny požadované parametry jako je správné testování hloubky, stínování, antialiasing, face culling, způsob osvětlení, základní texturování a zavolá se metoda `void Update(GLuint dt)`. Tato metoda, stejně jako ve všech ostatních scénách slouží k aktualizaci podle uplynulého času. Jedná se hlavně o aktualizaci polohy elektronů ve scéně.



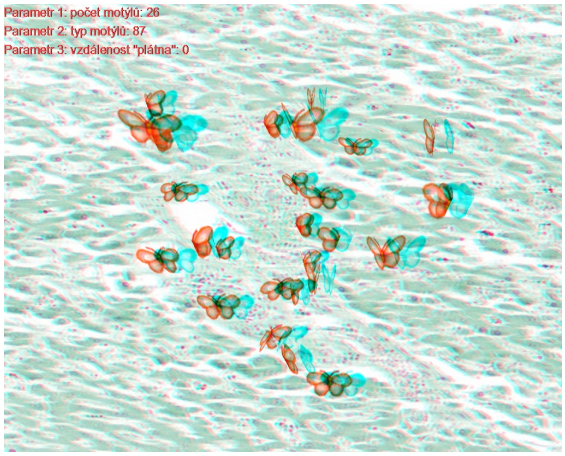
Ilustrace 18: AtomEffect

Podle vybraného způsobu zobrazení se nastaví stereoskopická projekce a začne se vykreslovat scéna voláním metody `void CreateEnvironment(int anaglyph)`. V této metodě se podle způsobu aplikace předchozí vrstvy volá jedna ze sedmi metod, které různě zobrazují texturu a následně se ještě volá

metoda pro rozmazání[9] je-li to třeba. V případě, že se má zobrazit celý vyrenderovaný atom jako textura na každé kuličce atomu, je to vyřešeno tak, že se zobrazený snímek také uloží a v příštím volání metody `draw()` se znovu použije jako textura a na konci se znovu uloží. Tím je dosaženo, že se celá scéna renderuje jen jednou, ale přitom na každé kuličce je zobrazena znovu a znovu. Rozmazávacího efektu je dosaženo tak, že vyrenderovaný snímek se uloží do pomocné textury, přepne se z perspektivní projekce do ortogonální a tato pomocná textura se několikrát vykreslí přes celý původní obrázek. Je však nutné, aby se vykreslovalo s nějakou úrovní průhlednosti a každé další vykreslení bylo ještě průhlednější než předchozí. Navíc s každým dalším vykreslením se zmenšuje výřez z pomocné textury, který se vykreslí.

6.6 Třída ButterflyEffect

Jelikož základ je u všech pěti scén stejný budeme se dále věnovat jen rozdílům. Jak již název napovídá, v této scéně jsme se snažili o vykreslování poloprůhledných motýlů před plátno. Jsou zde dvě pomocné struktury `Butterfly` a `Buttpoly`. První slouží k ukládání informací o jednom motýlovi, druhá je využita při řazení polygonů. Motýl se snaží doletět do určitého bodu v prostoru a jakmile se tam dostane vygeneruje se nový bod a opět se tam snaží dostat. K napodobení motýlího pohybu je využito funkce sinus tak, že se k jeho poloze ještě přičítá z času vypočtený offset ve všech třech osách. Jelikož motýli se jeví býti před plátnem, je nutné si dávat pozor, aby se při vygenerování nového bodu nedostával mimo záběr kamery. Pro efekt poloprůhlednoti motýlů se využívá blendovací funkce `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` s povoleným blendingem. Díky tomu se však musí nejprve vykreslit všechny polygony bez alfa kanálu a poté ostatní polygony seřadit podle vzdálenosti od kamery a nejprve vykreslovat ty nejvzdálenější. Pro řazení byla využita prioritní fronta, která je dle dokumentace implementována pomocí heap sortu, takže by měla být dostatečně efektivní. 3 základní textury motýlů byly převzaty z [15], byly však následně upraveny tak, aby motýli byli částečně průhlední a pomocí několika operací změn barvy vzniklo celkem 12 druhů motýlů.

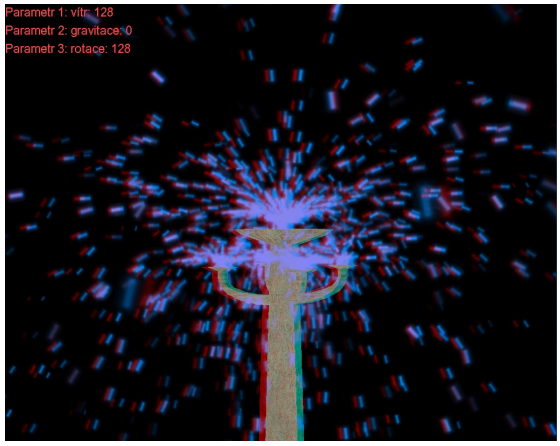


Ilustrace 19: ButterflyEffect

9 Tzv. Blur

6.7 Třída FountainEffect

V této scéně je zobrazena tryskající fontána. Každá kapka je řešena jako částice, která jakmile je vystříknuta žije vlastním životem a je ovlivňována jen časem, větrem a gravitací. Pro popis polohy částice bylo využito vztahu pro volný pád $s_0+\vec{v}_0\cdot t+0{,}5\cdot\vec{g}\cdot t^2$, kde s_0 je počáteční poloha, v_0 je počáteční rychlost, g je gravitační zrychlení a t je čas. Celkový počet částic je 2000, z toho 1000 připadá na střed fontány a 4x250 na kraje. Jelikož každá kapka je částečně průhledná a čím je starší tím je průhlednější, i zde bylo nutno seřadit všechny polygony před vykreslením. Tentokrát bylo přímo využito algoritmu sort z knihovny STL[10], který dle dokumentace má linearitmickou složitost, tedy N*logN. Při inicializaci se předpočítá stav fontány pro čas 2000ms kdy se scéna již jeví jako ustálená.



Ilustrace 20: FountainEffect

6.8 Třídy StarEffect a TextEffect

Jedná se o implementačně nejméně náročné třídy. V konstruktoru třídy `StarEffect` se nejprve inicializuje tvar hvězd, jejich počet a podobně jako u fontány i zde se předpočítá stav pro čas 10s. Jednotlivé parametry již byly popsány v návrhu. Za zmínku stojí, že pro vylepšení dojmu se hvězdy nepohybují rovně, ale každá trochu jiným směrem a navíc se sbíhají z krajů směrem ke kameře.

U třídy `TextEffect` je možné měnit barvu, rychlost a jeden ze šesti textů. Na této scéně je bohužel dobře vidět nevýhody anaglyfu, kdy čistě červený nebo čistě modrý/zelený objekt se nezobrazuje prostorově.

Ilustrace 22: TextEffect

Ilustrace 21: StarEffect

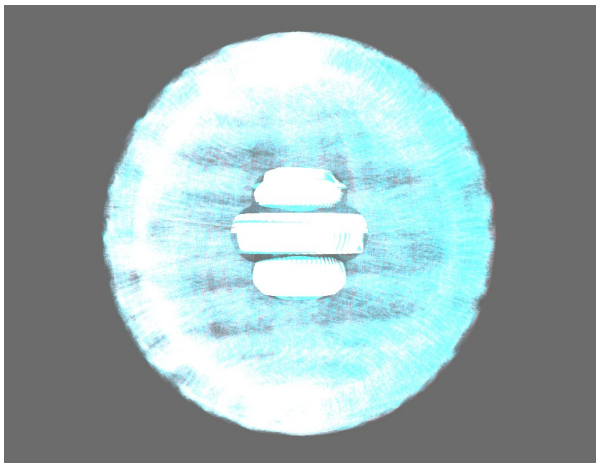
10 Standard Template Library – standartní knihovna dodávaná s jazykem C++

6.9 Pomocné rutiny

Mezi pomocné rutiny patří vytváření textur z obrázku nebo z textu a změna velikosti okna. Funkce pro vytváření textury z obrázku nejprve načte obrázek pomocí knihovny SDL_image, poté vytvoří pomocný surface do kterého se obrázek překopíruje. V tomto surface se přehodí řádky, jelikož SDL má počátek souřadného systému vlevo nahoře, zatímco OpenGL vlevo dole, nastaví se parametry OpenGL textury a překopírují se obrazová data do textury. Nakonec se uvolní všechna použitá paměť. Při vytváření textury z textu se vytvoří pomocný surface o předem dané velikosti, poté další 2 s textem různé barvy pomocí knihovny SDL_ttf a tento text se překopíruje do původního pomocného surface. Texty jsou však vzájemně posunuty o pár pixelu, čímž je dosaženo, že text má jakoby stín. Nakonec se z pomocného surface vytvoří textura a uvolní se paměť.

7 Testování

Vlastní testování probíhalo ve dvou fázích. Nejprve bylo vše testováno na mém vlastním PC, aby se odladilo co možná nejvíce chyb a teprve poté byla práce předána p. Farníkovi, aby to otestoval na DigitalSpotu. Jelikož počítač, na kterém se aplikace testovala[11], je o dost méně výkonný než počítač zabudovaný v DigitalSpotech, nebyl problém s tím, že by se vytvořila příliš náročná scéna a obraz by se v reálu sekal. Pouze u rozmazávacího efektu ve scéně s atomem bylo nutné volit mezi náročností a detailem vykreslení. Při testování bylo i různě experimentováno, většinou však výsledek nebyl použitelný ve stereoprojekci. Testovací textury pochází ze serveru [16] a jsou šířeny pod licencí GPL.



Ilustrace 23: příklad experimentování

Testování aplikace ukázalo několik faktů:

- anaglyf se jeví jako skutečně velmi nekvalitní metoda, je zde velmi výrazný ghosting jehož efekt se zhoršuje s rostoucí paralaxou.
- Ghosting a nevhodný poměr mezi konvergencí očí a vzdáleností na kterou se ostří jsou nejhorší věci, kterým se v reálu musíme co nejvíce vyhýbat.
- Na příliš průhledné věci je těžké zaostřit (např. někteří motýli nebo scéna s fontánou), je tedy vhodné u nich mít alespoň nějakou konturu, nebo jiné prvky na které se oči mohou chytit.
- Při použití anaglyfu red-cyan se osvědčily barvy obsahující zelenou, žlutou, fialovou a modrou, mnoho červené se jevilo skoro až jako černá, velmi však záleží na barvě pozadí.
- Dost efektů v klasickém zobrazení vypadá lépe než při stereoprojekci, platí to však i obráceně.
- DigitalSpoty mají funkci Keystoning, pomocí které je možné obrazy seřídit tak, aby se přesně překrývaly ačkoli každé světlo promítá z jiného úhlu – je vidět na fotografiích v příloze.
- Ne každý stejně vnímá danou scénu a každému trvá trochu jinak dlouho než si jeho oči přivyknou anaglyfu. Každý také jinak vnímá ghosting a někoho ruší méně, někoho více.
- Jednoznačně nejlepší ohlasy od přátel při testování na svém počítači byly při scéně s motýly, většina z nich se je pokusila uchopit do ruky.

11 CPU – Intel Core2Duo 2,0GHz – 800MHz FSB, paměť – 2GB RAM, grafická karta – GeForce 8600M GT

8 Závěr

V této práci jsme se převážně zabývali způsobem jak vytvořit stereoskopickou projekci, co ji ovlivňuje a jak ji dále vylepšit a bylo popsáno několik běžně používaných technologií. Dále pak byly teoreticky popsány knihovny využité při návrhu a implementaci práce. Tato práce navazuje na již existující vývoj osvětlovací techniky DigitalSpot.

Z praktického hlediska, o čemž svědčí i spolupráce s firmou ROBE, jsem navrhl a následně implementoval aplikaci pro zobrazování 3D objektů pro prostorové vidění. Díky použitým knihovnám je možné aplikaci přeložit jak ve Windows tak i v Linuxu. Získal jsem obstojné znalosti z počítačové grafiky a stále více populární stereo projekce.

Do budoucna by bylo možné vytvořit podporu pro načítání dat z dalších formátů jako jsou .x (DirectX 3D Model), .blend (Blender), .ctm (OpenCTM) a dalších. Dále by bylo možné aplikaci rozšířit o spoustu grafických efektů a nových scén.

Seznam použitých zkratek

2D (two dimension, dvě dimenze) – symbolizuje plošné nebo také dvourozměrné vyjádření prostoru či objektu

3D (three dimension, tři dimenze) – symbolizuje prostorové nebo také třírozměrné vyjádření prostoru či objektu

SDL (Simple Direct-Media Layer) – knihovna používaná pro snadnou komunikaci s operačním systémem

LCD (Liquid-crystal display, displej z tekutých krystalů) – tenké ploché zobrazovací zařízení

OpenGL (Open Graphics Library, volná grafická knihovna) – otevřená definice pro aplikační rozhraní na grafický hardware

GLUT (OpenGL Utility Toolkit, pomocný toolkit k OpenGL) – doplněk ke grafické knihovně OpenGL

GLU (OpenGL Utility Library, pomocná knihovna k OpenGL) – doplněk ke grafické knihovně OpenGL

GLUI (OpenGL User Interface Library, knihovna pro uživatelské rozhraní k OpenGL) – doplněk ke grafické knihovně OpenGL

GPL (General Public License, všeobecná veřejná licence) – licence svobodného softwaru

LGPL (Lesser General Public License, menší obecná veřejná licence) – licence svobodného softwaru, je tolerantnější než GPL

TTF (TrueType Font) – standard pro popis vektorových písem

RTF (Rich Text Format, rozsáhlý textový formát) – formát souborů pro uložení textu, který obsahuje velké množství formátovacích příkazů

PNG (Portable Network Graphics, síťový přenosný grafický formát) – formát pro ukládání obrazu s bezeztrátovou kompresí

JPG (Joint Photographic Experts Group) – formát pro ukládání obrazu se ztrátovou kompresí

GIF (Graphics Interchange Format, grafický výměnný formát) – formát pro ukládání obrazu s bezeztrátovou kompresí, umožňuje jednoduché animace

API (Application Programming Interface, aplikační programovací rozhraní) – rozhraní pro programování aplikací, jde o sbírku funkcí a/nebo tříd, které může programátor využívat

3DS (3D-Studio) – formát pro ukládání 3D objektů

MOC (Meta-Object Compiler, kompilátor pomocných objektů) – slouží pro překlad zdrojovych souboru, jenž obsahují klíčové slova `signal`, `slot`, `emit` a případně další pomocná makra

DPS (Dynamic Property System, systém dynamických vlastností) – umožnuje v objektech používat properties

RTTI (Run Time Type Information, informace o typu za běhu programu) – systém, který udržuje data o typu objektu v paměti za běhu aplikace

XML (Extensible Markup Language, rozšiřitelný značkovací jazyk) – velice oblíbený formát souborů pro ukládání strukturovaných dat

DMX – standard pro digitální komunikační sítě, běžně používaný pro ovládání osvětlovací techniky

LED (Light-Emitting Diode, světlo emitující dioda) – polovodičová součástka obsahující PN přechod, protéká-li jí proud, vyzařuje světlo o určité vlnové délce

ABC (Abstract Base Class, abstraktní bázová třída) – třída objektově orientovaného jazyka (C++), z takovéto třídy nelze vytvořit objekt, lze z ní však odvodit jinou třídu

STL (Standard Temlate Library, standardní knihovna šablon) – knihovna, jenž se běžně dodává spolu s jazykem C++

Literatura

[1] Kognitivní server, Univerzita Hradec Králové – http://cogn.uhk.cz. [online]. [cit. 19.4.2010]. Dostupné na URL: http://fim.uhk.cz/cogn/?Module=dictionary&TypeSearch=1&Str=binokul%E1rn%ED+disparita

[2] ČÍŽEK, P.: *Prostorové zobrazování*. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. 2005.

[3] RICARDO JOSÉ TEXEIRA DOS SANTOS: *Scaling of Rendered Stereoscopic Scenes.* Master Thesis Report. University of West Bohemia in Pilsen. Czech Republic. 2005.

[4] fotoklub FOTOKAMZÍCI: *Stereografie – třetí rozměr vefotografii*. [online]. rev. 19. 3. 2008. [cit. 22.4.2010]. Dostupné na URL: http://www.fotokamzici.com/aktuality/stereofotografie-treti-rozmer-ve-fotografii/222

[5] NAČASOVÁ, H.: *Dvojstředové promítání, anaglyfy.* Diplomová práce. Masarykova univerzita v Brně, Přírodovědecká fakulta, Ústav matematiky a statistiky. 2007.

[6] Horejš, M. (editor): *REBURBER. Zpravodaj Národního technického muzea.* Národní technické muzeum. 2009.

[7] WOODS A. J., TAN S. S. L.: *Characterising Sources of Ghosting in Time-Sequential Stereoscopic Video Displays*. Centre for Marine Science and Technology, Curtin University of Technology. 2002.

[8] WOODS A. J., ROURKE T.: *Ghosting in Anaglyphic Stereoscopic Images*. Centre for Marine Science and Technology, Curtin University of Technology. 2004.

[9] BOURKE P.: *Calculating Stereo Pairs.* [online]. rev. July 1999. [cit. 25.4.2010]. Dostupné na URL: http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/stereographics/stereorender/

[10] Games for Windows®: *Microsoft® DirectX®11*. [online]. [cit. 26.4.2010]. Dostupné na URL: http://www.microsoft.com/games/en-US/aboutGFW/pages/directx.aspx

[11] The Free OpenGL Utility Toolkit: *freeglut*. [online]. rev. 27 November 2009. [cit. 27.4.2010]. Dostupné na URL: http://freeglut.sourceforge.net

[12] Simple DirectMedia Layer: *Libraries*. [online]. [cit. 27.4.2010]. Dostupné na URL: http://www.libsdl.org/libraries.php

[13] lib3ds: *lib3ds is an overall software library for managing 3D-Studio Release 3 and 4 ".3DS" files.* [online]. [cit. 28.4.2010]. Dostupné na URL: http://code.google.com/p/lib3ds/

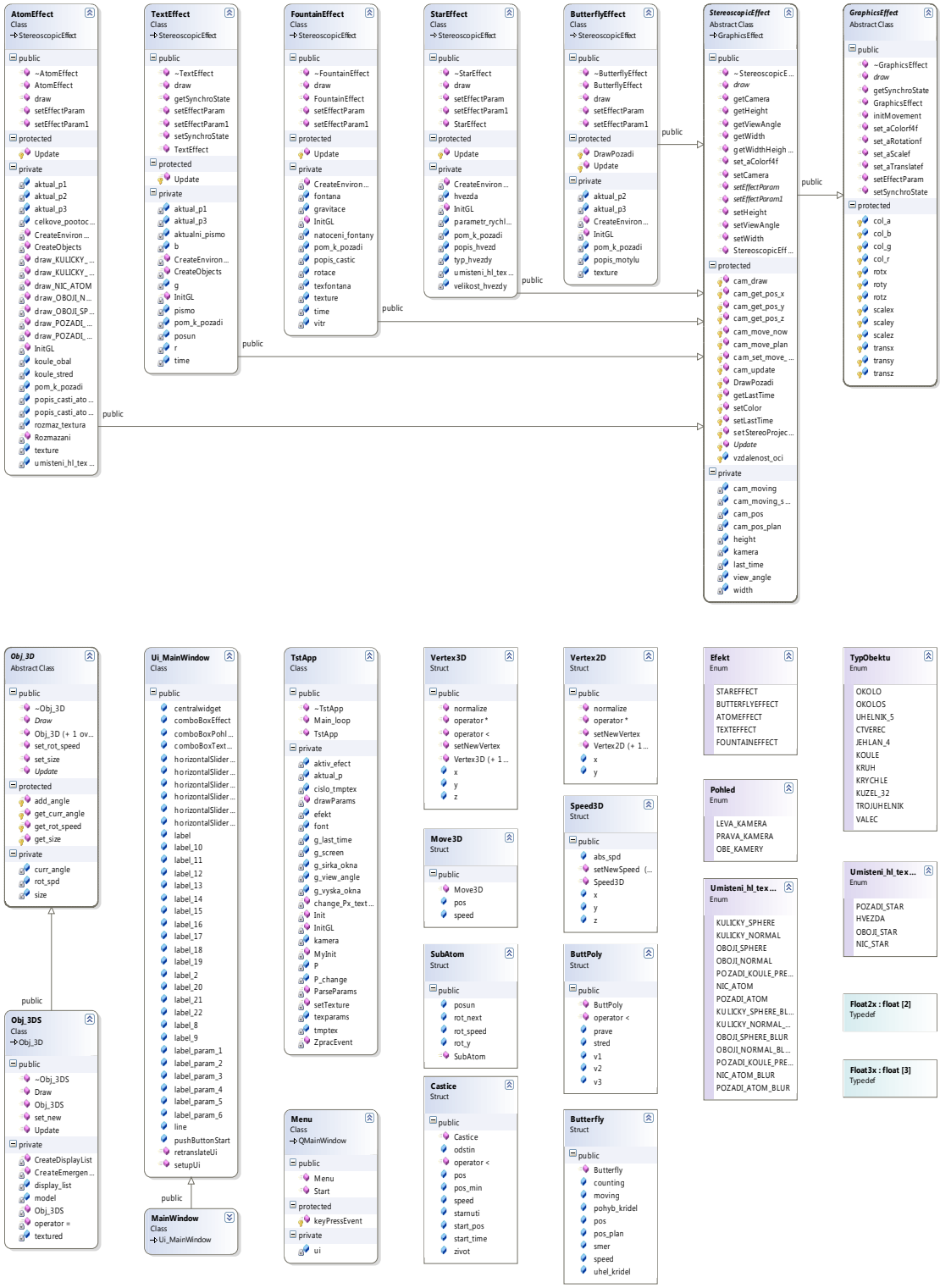
[14] Department of computer science: *Matrix Transformation*. [online]. [cit. 8.5.2010]. Dostupné na URL: http://www.cs.uregina.ca/Links/class-info/405/WWW/Lab3/

[15] NeonHelium Production: *Lesson 38*. [online]. [cit. 15.5.2010]. Dostupné na URL: http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=38

[16] Burningwell.org: *Public domain image source*. [online]. rev. 22. Apr 2006. [cit. 15.5.2010]. Dostupné na URL: http://www.burningwell.org/

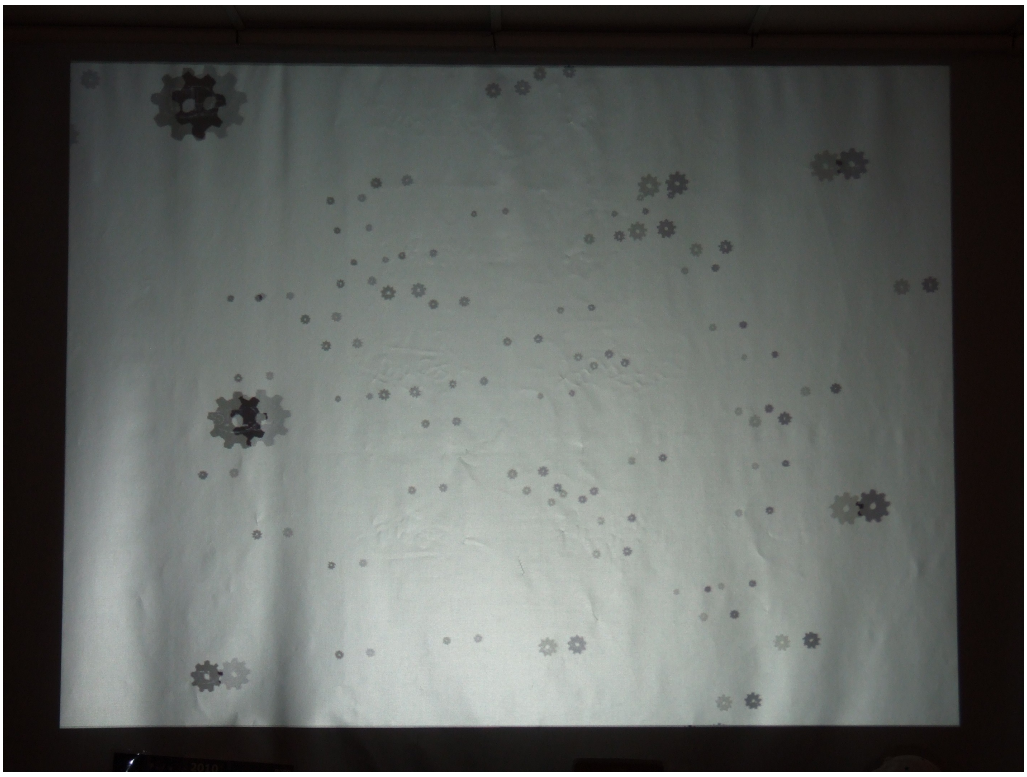
Seznam příloh

Příloha A Celý diagram tříd včetně všech datových typů



Ilustrace 24: celý diagram tříd a pomocných struktur

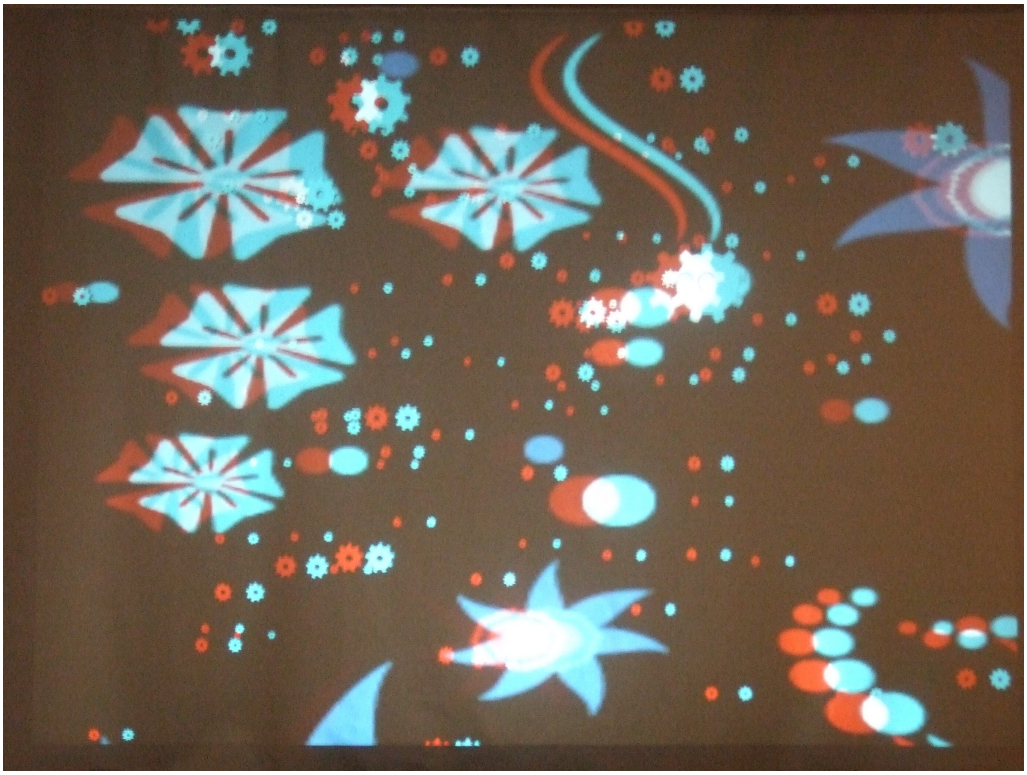
Příloha B Fotografie z testování



Ilustrace 25: detail plátna s ozubenými koly

Ilustrace 26: ukázka anaglyfu

Ilustrace 27: rotující atom: díky špatným barvám a malé paralaxe nevýrazný stereoskopický efekt



Ilustrace 28: stejné jako předchozí jen s anaglyfem a texturou v hloubce

Ilustrace 29: scéna s fontánou

Ilustrace 30: scéna s motýly

Příloha C Návod ke kompilaci

Tento návod popisuje jen jednu z možných variant kompilace ve Windows a v Linuxu. V případě, že se nepodaří kompilaci provést je možné spustit již zkompilovanou verzi na CD.

Postup ve Windows:

Nainstalujte MinGW (na adrese http://sourceforge.net/projects/mingw/files/ je možné stáhnout instalátor), při instalaci vyberte alespoň položky g++ compiler a MinGW Make. Po instalaci přidejte do systémové cesty Path: ";<cesta k MinGW>\bin", (v některých verzích Windows je třeba restart počítače). V tomto adresáři bin se nachází soubor mingw32-make.exe, přejmenujte ho na make.exe anebo přidejte systémovou proměnnou tak, aby se při příkazu make spouštělo mingw32-make. Tím je vyřešena instalace překladače a programu make, nyní ještě stáhnout knihovny SDL.

Knihovna SDL je dostupná na adrese http://www.libsdl.org/release/SDL-devel-1.2.14-mingw32.tar.gz (rozbalte a výsledek překopírujte do dříve nainstalovaného MinGW, tak aby se adresáře bin, include a lib spojily).

Knihovna SDL_image je na adrese http://www.libsdl.org/projects/SDL_image/release/SDL_image-devel-1.2.10-VC.zip (soubor SDL_image.h zkopírovat do složky "<cesta k MinGW>\include\SDL", dll knihovny do adresáře bin, lib knihovnu do adresáře lib).

Knihovna SDL_ttf je na adrese http://www.libsdl.org/projects/SDL_ttf/release/SDL_ttf-devel-2.0.9-VC8.zip (opět rozbalit a soubor SDL_ttf.h zkopírovat do složky "<cesta k MinGW>\include\SDL", dll knihovny do adresáře bin, lib knihovnu do lib).

Tím je vyřešena hlavní část, pokud při testování chcete využít i úvodního menu (není nutnost), je zapotřebí ještě doinstalovat knihovnu Qt verze 4 a vyšší. To je možné na adrese http://qt.nokia.com/downloads, zvolit možnost LGPL a při instalaci postupovat dle instrukci.

Jakmile jsou všechny knihovny nainstalovány stačí se přesunout do kořenové složky v bakalářské práci a napsat make -f Makefile.win. Výsledek bude umístěn ve složce release_win.

Postup v Linuxu:

Zde se jako nejlepší řešení jeví přes správce balíčků nainstalovat všechny knihovny SDL, překladač a Qt4. V případě, že to z nějakého důvodu není možné, na adresách:

- http://gcc.gnu.org/install/index.html (návod na instalaci GCC)
- http://www.libsdl.org/download-1.2.php
- http://www.libsdl.org/projects/SDL_image/
- http://www.libsdl.org/projects/SDL_ttf/
- http://qt.nokia.com/downloads

je možné stáhnout požadovaný software (osobně však netestováno). Jakmile jsou všechny knihovny připravené stačí se přesunout do kořenové složky v bakalářské práci a napsat make -f Makefile.linux. Výsledek bude umístěn ve složce release_linux.